DESX.IO
FUTURE OF TOKEN

WHITE PAPER

Content

1. Greeting

Warm greetings to all members, partners, and investors at DES X . Exchange.
We greet you with hope and excitement as we join you at DES X - a pioneering and groundbreaking Decentralized Exchange.

We live in an era full of challenges as well as opportunities. Financial values are undergoing a dramatic transformation, and DES X was born as a powerful attempt to reshape the way we think about trading, investing, and creating value. To provide a transparent, fair, and secure trading environment, DES X is committed to bringing innovation and significant difference.

We build DES X based on advanced technology and deep knowledge, to create a dynamic and flexible trading space. Convenience, speed, and accessibility are our strengths, enabling investors and market participants to make the most of every opportunity.

We know that the success of DES X cannot be achieved without the contribution and support of all members. With unanimous participation, we can build a solid and prosperous community, where creativity and possibilities for growth are unlimited.

We are pleased to welcome you into DES X, where innovation, interaction, and value creation are at the heart. Let's give wings to a bright and sustainable financial future.

2. DES-X - Decentralized Finance Platform With Comprehensive Advantages

Welcome to the DES-X Exchange, which has everything you need to participate in the decentralized finance revolution. With a unique combination of trading, optimization, aggregation, market value, security, connectivity, privacy, visibility, distributed security, and community value, DES-X delivers a whole new experience for participants.

Trading: Explore the Modern World of Trading.
DES-X is a promising cryptocurrency trading platform that allows you to participate in the rapidly growing market with modern and quality trading tools.

Optimization: Leveraging the Benefits of Performance Optimization
With DES-X, you not only trade but also optimize the performance of your mining equipment, helping you get the most out of your cryptocurrency mining.

General: Connecting With Different Exchanges.

The aggregation of information from different exchanges helps you to track and manage your assets easily and efficiently.

Market Value: Understanding and Capitalizing on Market Volatility.
DES-X gives you access to up-to-date market value information, helping you make smart trading decisions and predict market trends.

Privacy: Protect Your Privacy.
Privacy is a priority at DES-X, helping you to engage in cryptocurrency trading and mining safely and securely.

Connect Trade Anytime, Anywhere.
With DES-X, you can connect and trade cryptocurrencies anytime, anywhere, via mobile devices or personal computers.
Security: Protect Your Property and Data.
DES-X is committed to ensuring the security of assets and data during trading and mining.

Display: Easy Monitoring And Management.
Thanks to the intuitive information display, you can track and manage your assets easily and efficiently.

Distribution Security: Securing the Distribution Process.
DES-X ensures safety and transparency in the distribution of cryptocurrencies, ensuring that all transactions are carried out securely.

Community Values: Joining and Contributing to the Financial Future.
By participating in DES-X, you contribute to building a free financial environment

3. What is DES-X?

Welcome to the world of DES X - a state-of-the-art Decentralized Exchange System (DeS-X) set to revolutionize the cryptocurrency trading landscape. In this article, we will dive into the innovative features and benefits offered by the DES X platform, highlighting how DeS-X technology is changing the way users securely trade cryptocurrencies. and easy.

3.1: Learn about DES X - DeS-X Technology.

A brief overview of the DES X platform and its core principles.

Explanation of DeS-X technology, highlighting the decentralized nature of the exchange and no intermediaries.

Discuss the advantages of DeS-X, such as enhanced security, privacy, and money control.

3.2: Seamless Transactions on DES X.

User-friendly interface and easy navigation for both beginners and experienced traders.

Integrate with popular wallets for convenient transaction and fund management.

Fulfill orders in a peer-to-peer manner for fast and efficient transactions.

3.3: Security measures and reliable transactions.

Smart contract technology ensures safe and reliable transactions without depending on third parties.

User funds protection mechanism to protect against potential vulnerabilities.

Eliminates a single point of failure, minimizing the risk of hacking or system crashes.

3.4: Advanced trading features.

Advanced charting and analysis tools for comprehensive market analysis.

Limits and market orders for flexible trading strategies.

Automated trading options for algorithmic traders.

3.5: Liquidity on DES X.

How DES X addresses the liquidity challenges that decentralized exchanges often face.

Incentives for liquidity providers to foster a vibrant trading ecosystem.

Integrate with other liquidity pools to enhance trading options.

3.6: DES Layer - Token of the DES X . Ecosystem.

Introduction to DES. exchange, the native utility token of the DES X platform.

Utilities and benefits of DES. exchange in the ecosystem, such as reducing transaction fees and voting rights.

Details of the token distribution model and future plans for DES.

Conclude

The DES X platform, powered by DeS-X technology, marks an important step towards a decentralized future for cryptocurrency trading. By prioritizing security, user control, and a seamless trading experience, DES X is positioning itself as a leading player in the ever-evolving crypto exchange landscape. Embrace the power of DES X and join the DeS-X revolution today!

4. DES stands for "Decentralized Trust Service".

Decentralized: Refers to the underlying architecture of an exchange, which operates on a decentralized network, similar to a DEX, to provide enhanced security and user control.

Escrow Services: In the context of cryptocurrency exchanges, escrow services ensure safe and secure money handling in peerto-peer transactions. It acts as a trusted intermediary that holds the funds until both parties fulfill their obligations in the transaction.

DES is a new type of cryptocurrency exchange that combines the advantages of both centralized and decentralized exchanges while introducing a secure escrow service to facilitate peer-topeer transactions. It aims to provide users with a seamless and reliable platform for trading digital assets with reduced counterparty risk and improved transaction security. With DES, users can trade directly with each other while enjoying the protection and trust provided by the decentralized escrow service.

5. Future of Cryptocurrency Trading: DESAdvantages over CEX and DEX.

Cryptocurrency trading has evolved over the years, with centralized exchanges (CEX) and decentralized exchanges (DEX) dominating the landscape. However, a new paradigm shift is taking place with the emergence of "Decentralized Escrow Service" (DES) exchanges. In this article, we will explore the unique advantages of DES over CEX and DEX, focusing on verified liquidity and enhanced protection of user funds against potential scams.

5.1: Verified liquidity on DES.

The DES platform uses innovative techniques to verify liquidity and ensure a smooth trading experience.

Integrate with multiple liquidity providers and external liquidity pools to enhance trading options.

Users can access real-time liquidity data, allowing them to make informed trading decisions.

DES eliminates liquidity bottlenecks, promoting a dynamic and liquid trading environment.

5.2: Advanced User Fund Protection.

DES exchange implements advanced security measures to protect user funds from malicious intent.

Decentralized escrow services act as a secure intermediary, holding trust funds until trading conditions are met.

Smart contract technology ensures automatic fund release and anti-counterfeiting after successful transaction completion.

Users have full control over their funds, minimizing the risk of exchanges holding users' funds and facing potential scams.

5.3: Minimize the risk of running away.

Traditional CEXs can run the risk of exiting or fleeing with user funds due to centralized control.

DEXs, while decentralized, may lack robust measures to prevent fraudulent activities.

The DES platform creates a balance, using decentralized features while providing the security of an escrow service.

Money is kept in smart contracts, reducing the likelihood of malicious activities and improving user trust.

5.4: Trust and transparency.

The DES exchange prioritizes transparency, providing users with visibility into their trading activities.

Immutable on-chain transactions allow users to track their transactions and verify their execution.

Open source and smart contract audits ensure an extra layer of security and trust.

5.5: Seamless peer-to-peer transactions.

DES facilitates peer-to-peer transactions without intermediaries, enhancing privacy and control.

Users retain ownership of their private keys, eliminating the risk of exchange custody control.

Direct transactions promote faster payments and lower transaction fees.

Conclude

As the crypto space continues to grow, DES exchanges are emerging as a revolutionary solution to address the shortcomings

of both CEX and DEX. Verified liquidity, enhanced protection of user funds, and reduced risk of fraud make DES an attractive choice for crypto traders looking for a secure, transparent and secure trading experience. decentralized. By leveraging decentralized escrow services, DES platforms are shaping the future of cryptocurrency trading and empowering users to take full control of their digital assets. As the industry adopts this innovative approach, DES is poised to revolutionize the crypto exchange landscape.

6. Optimize your trading.

Provide excellent service experience quickly - not as complicated as traditional DEX solutions. Lower fees, and fast transaction processing for absolute safety and security.

6.1: Continuous integration of anti-bot technology and absolute user protection.

One of the core factors that make DESX stand out is the continuous integration of advanced anti-bot technology, along with a natural user protection system. This ensures that every

transaction is done by real users while preventing fraudulent activity and adversely affecting market participants

6.2: Super Fast Transactions and Transaction Fee Refunds Through DESX.

Time is the deciding factor in trading, and DESX understands this. The transaction speed on DESX is outstanding, helping you to execute transactions quickly and efficiently. The special thing is that you can refund transaction fees through the use of DESX tokens, creating an interactive connection between the platform and users

6.3: Use AI in trading, minimizing slippage.

With the integration of artificial intelligence (AI), DESX not only creates a high-speed trading environment but also minimizes slippage. The use of AI helps predict and adapt to market changes, ensuring that you get the best price on every trade

Experience Super Fast Trading with DESX: The Perfect Combination of Technology and Efficiency

No more delays or hassles from traditional DEX solutions, DESX delivers the super-fast trading experience you've always come to expect. With the unique integration of anti-bot technologies, protect real users.

7. DESX Launchpad.

A curated token launch platform or we refer to it as a crypto launchpad, provides investors with a secure and efficient way to invest in promising projects and benefit from the lucrative IDO and ICO market.

7.1: Project evaluation.

The smart choice for credibility with the DESX launcher, every project is carefully vetted before being put on the platform. This ensures that only projects with high potential and reputation can access capital from the investment community, creating credibility and transparency in the investment process.

7.2: Multi-Chain Support.

Scalability and Flexible Integration The DESX launcher is not limited by a single blockchain. The ability to support multichain allows the platform to expand and integrate flexibly into many different environments, making it easy for projects and investors to access and participate.

7.3: Generate Tokens Directly from the Platform.

Friendly and Time-Saving By allowing token generation directly from the platform, DESX brings friendliness and time-saving to projects and investors. No need to go through complicated steps, the token generation process becomes easy and fast.

7.4: Supporting Crypto Gaming, NFT, and Metaverse Projects.

Investing in the Future Besides the mainstream market, DESX also focuses on supporting promising crypto gaming, NFT, and metaverse projects. This provides an opportunity to invest in the future of technology and digital art, creating diversity and great potential for the investment community.

8. FOR DEVELOPERS.

Why join the DES development team?

1. Discover the Essence of Technology: At DES, you will have the opportunity to study and work with the most advanced technology in the field of decentralized trading. This not only gives you access to fresh knowledge but also fosters creativity in system development and optimization.
2. Creating a New Financial Environment: By joining us, you will contribute to building a transparent, safe, and efficient trading environment. Your work will help create the necessary infrastructure for future market participants.
3. Challenges and Opportunities: Joining the development team of DES allows you to challenge and grow. You will face complex problems, discover new solutions, and go further in your development career.
4. Collaborate and Learn: Our development team is not only a workplace but also a community of collaboration and learning.

You will have the opportunity to exchange knowledge, share ideas and learn from excellent colleagues.

How to Protect DES Against Exhaustive Key Search (An Analysis of DESX)*

JOE KILIAN† PHILLIP ROGAWAY‡

February 2, 2000

Abstract

The block cipher DESX is defined by $\mathrm{DESX}_{k.k1.k2}(x) = k2 \oplus \mathrm{DES}_k(k1 \oplus x)$, where $\oplus$ denotes bitwise exclusive-or. This construction was first suggested by Rivest as a computationally-cheap way to protect DES against exhaustive key-search attacks. This paper proves, in a formal model, that the DESX construction is sound. We show that, when F is an idealized block cipher, $FX_{k.k1.k2}(x) = k2 \oplus F_k(k1 \oplus x)$ is substantially more resistant to key search than is F. In fact, our analysis says that FX has an effective key length of at least $\kappa + n - 1 - \lg m$ bits, where κ is the key length of F, n is the block length, and m bounds the number of $\langle x,\ FX_K(x)\rangle$ pairs the adversary can obtain.

Key words: DESX, DES, Key search, Cryptanalysis, Export controls.

1 Introduction

With its 56-bit keys, the susceptibility of DES to exhaustive key search has been a concern and a complaint since the cipher was first made public; see, for example, [6]. The problem has escalated to the point that the Electronic Frontier Foundation has now built a DES cracking machine, at a cost of less than 250,000 USD, that can find the right key in about three days.

There have been many approaches suggested for reducing DES's vulnerability to exhaustive key search. One is to construct a DES-based block cipher which employs a longer key. Triple DES (typically in "EDE mode") is the best-known algorithm in this vein. It seems to be quite secure, but efficiency considerations make triple DES a rather painful way to solve the exhaustive key-search problem. Specifically, triple-DES encryption/decryption requires multiple DES encryptions/decryptions. This paper analyzes a much cheaper alternative.

Rivest [13] proposes an extension of DES, called DESX, defined by

$$\mathrm{DESX}_{k.k1.k2}(x) = k2 \oplus \mathrm{DES}_k(k1 \oplus x).$$

*An earlier version of this paper appears in [11].

† NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA. E-mail: joe@research.nj.nec.com

‡ Department of Computer Science, University of California at Davis, Davis, CA 95616, USA. E-mail: rogaway@cs.ucdavis.edu

The key $K = k.k1.k2$ (here, . denotes concatenation) is now $56+64+64 = 184$ bits. Compatibility with DES is maintained by setting $k1 = k2 = 0^{64}$. Existing DES CBC hardware can be gainfully employed by first masking the plaintext, computing the DES CBC, and then masking the ciphertext. Most significantly, DESX has hardly any computational overhead over ordinary DES. Yet, somehow, DESX seems no longer susceptible to brute-force attacks of anything near 2^{56} time.

It is unintuitive that one should be able to substantially increase the difficulty of key search by something as simple as a couple of XORs. Yet working with the DESX definition for a while will convince the reader that undoing their effect is not so easy.

Does the "DESX trick" really work to improve the strength of DES against exhaustive key search? We give a strong positive result showing that it does.

1.1 Our model

Key-search strategies disregard the algebraic or cryptanalytic specifics of a cipher and instead treat it as a black-box transformation. Key-search strategies can be quite sophisticated; recent work by [16] is an example. We want a model generous enough to permit sophisticated key-search strategies, but restricted enough to permit *only* strategies that should be regarded as key search. We accomplish this as follows.

Let κ be the key length for a block cipher and let n be its block length. We model an *ideal* block cipher with these parameters as a *random* map $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ subject to the constraint that, for every key $k \in \{0,1\}^\kappa$, $F(k,\cdot)$ is a permutation on $\{0,1\}^n$. A key-search adversary A is an algorithm that is given the following two oracles:

- An F oracle that on input (k,x) returns $F(k,x)$ and
- An F^{-1} oracle that on input (k,y) returns $F^{-1}(k,y)$.

Here, $F^{-1}(k,y)$ denotes the unique point x such that $F(k,x) = y$.

A *generic key-search adversary* tries to perform some cryptanalytic task (to be specified) that depends on F. She may perform arbitrary computations, using unbounded amounts of time and space, but her only access to F is via the F/F^{-1} oracles. We analyze the adversary's rate of success in performing her cryptanalytic task as a function of the number of accesses she makes to the F/F^{-1} oracles.

To apply the above framework to DESX, we first generalize the DESX construction. Given any block cipher F we define $FX : \{0,1\}^{\kappa+2n} \times \{0,1\}^n \to \{0,1\}^n$ by

$$FX(k.k1.k2,\ x) = k2 \oplus F(k, k1 \oplus x).$$

For both F and FX we shall sometimes write their first argument (the key) as a subscript, $F_k(x)$ and $FX_K(x)$, where $K = k.k1.k2$. In this notation, F_k may be thought of as a permutation chosen from a family of (random) permutations that is indexed by k.

To investigate the strength of FX against key search we consider a generic key-search adversary A with oracles for F and F^{-1}, and determine how well A can play the following "FX–or–π" game.

A is given an "encryption oracle" E that has been randomly chosen in one of two ways (each with probability 0.5):

- A string $K \in \{0,1\}^{\kappa+2n}$ is chosen at random and $E(x) = FX_K(x)$, or
- A random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ is selected and $E(x) = \pi(x)$.

A must guess which way E was chosen. The adversary "wins" the game if it guesses correctly with probability significantly greater than 0.5. The FX construction "works" if the resources needed to do a good job in winning the above game are substantially *greater* than the resources that suffice to break F.

As an example of a generic key-search attack, consider the weakened form of DESX, denoted DESW, in which k_1 is always set to $0^{|k_1|}$; that is,

$$\mathrm{DESW}_{k.k2}(x) = k2 \oplus \mathrm{DES}_k(x).$$

It is possible to mount a generic key-search attack DESW as follows. Given k and $\mathrm{DESW}_{k.k_2}(x)$ for an arbitrary x, one can compute $k_2 = \mathrm{DESW}_{k.k_2}(x) \oplus \mathrm{DES}_k(x)$. Thus, one can go through all possible keys k, compute the full key $k.k_2$, and test with high confidence whether $k.k_2$ is correct (given values of $\mathrm{DES}_{k.k_2}(y)$ for a couple of random y-values). Hence, DESW is no stronger than DES against generic key-search attacks. Similarly, if k_2 is always set to $0^{|k_2|}$, there is no significant improvement over DES, as long as two or three plaintext-ciphertext pairs are known. (There may be marginal benefits if only a single plaintext-ciphertext pair is known, or for ciphertext-only attacks, but these are comparatively small improvements.) It is the combination of the two XOR operations that give DESX its superior resistance to generic key-search attacks.

1.2 Our main result

We show that if generic key-search adversary A can make only a "reasonable" number to queries to her encryption oracle E, then A must ask an excessive number of F/F^{-1} queries in the FX-or-π game, and therefore A must run for an excessively long time. More specifically, we prove the following. Let m bound the number of $\langle x, FX_K(x)\rangle$ pairs that the adversary can obtain. (This number is usually under the control of the security architect, not the adversary.) Suppose the adversary makes at most t queries to her F/F^{-1} oracles. (This number is usually under the control of the adversary, not the security architect.) Then the adversary's advantage over random guessing (i.e., the difference between its success and failure probabilities) in winning the FX–or–π game is at most $mt \cdot 2^{-\kappa-n+1}$. In other words, the adversary's advantage is at most $t \cdot 2^{-\kappa-n+1+\lg m}$, so the effective key length of FX, with respect to key search, is at least $\kappa + n - 1 - \lg m$ bits.

To understand the above formula, consider a block cipher F with 55-bit keys and a 64-bit block size.[1] Suppose key-search adversary A attacks FX and in the course of attack able to obtain up to $m = 2^{30}$ blocks of enciphered data. Suppose A runs in time at most T. Then A has advantage of at most $T \cdot 2^{-55-64+30+1} = T \cdot 2^{-88}$ to just guess whether the enciphered data really *was* produced by FX, and not a random permutation. A more detailed discussion of out main theorem is given in Section 4.

[1] Why we use 55 and not 56 is explained in the discussion in Section 4.

Because our main result indicates the infeasibility of key search even when we ignore the adversary's space requirement, this "omission" only strengthens what we are saying. Similarly, "good" adversaries may, necessarily, use an amount of time, T, which far exceeds their number of F/F^{-1} queries, t. So focusing on the query complexity makes our results all the more meaningful. Likewise, the weakness of the adversary's goal only strengthens the lower bound.

1.3 Related work

Even and Mansour [8] construct a block cipher $PX : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ from a random permutation $P : \{0,1\}^n \to \{0,1\}^n$ by $PX_{k1.k2}(x) = k2 \oplus P(k1 \oplus x)$. Clearly this is a special case of the FX construction, where $\kappa = 0$. While their motivation for looking at PX was quite different from our reasons to investigate FX, our model and methods are, in fact, quite similar. Our main result can be seen as a natural extension of their work.

The modeling of a block cipher by a family of random permutations has its roots in [15].

Ron Rivest invented DESX by May of 1984, but never described the scheme in any conference or journal paper [13]. DESX was implemented within products of RSA Data Security, Inc., and is described in the documentation for these products [14]. DESX has also been described at conferences organized by RSA DSI, including [18].

Encryption methods similar to DESX have been invented independently. Blaze [3] describes a DES mode of operation in which the ith block of plaintext, x_i, is encrypted using 112-bit key $k.k1$ by $E_{k.k1}(x_i) = s_i \oplus \mathrm{DES}_k(s_i \oplus x)$, where $s_1 s_2 \cdots$ is a stream of bits generated from $k1$ by, say, $s_i = \mathrm{DES}^{(i)}_{k1}(0^{64})$. Here $\mathrm{DES}^{(i)}$ denotes the i-th iterate of DES.

Many authors have suggested methods to increase the strength of DES by changing its internal structure. Biham and Biryukov [1] give ways to modify DES to use key-dependent S-boxes. Their suggestions improve the cipher's strength against differential, linear, and improved Davies' attacks, as well as exhaustive key search. Ciphers constructed using their ideas can exploit existing hardware exactly in those cases where the hardware allows the user to substitute his own S-boxes in place of the standard ones.

1.4 Discussion

UNDERSTANDING OUR RESULT. It may be hard to understand the ramifications of our main theorem, thinking it means more or less than it does. DES, of course, is not a family of random permutations, and we can *not* conclude from our theorem that there does not exist a reasonable machine M which breaks DESX in say, 2^{60} steps, given just a handful of ⟨plaintext, ciphertext⟩ pairs. What we can say is that such a machine would have to exploit structural properties of DES; it couldn't get away with treating DES as a black-box transformation. This contrasts with the sort of machines which have been suggested in the past for doing brute-force attack: they *do* treat the underlying cipher as a black-box transformation.

We note that while remarkable theoretical progress has been made on the linear and differential cryptanalysis of DES (see [2, 12]), thus far these attacks require an impractically large number of plaintext-ciphertext pairs. To date, the only published practical attacks against DES remain of

Because our main result indicates the infeasibility of key search even when we ignore the adversary's space requirement, this "omission" only strengthens what we are saying. Similarly, "good" adversaries may, necessarily, use an amount of time, T, which far exceeds their number of F/F^{-1} queries, t. So focusing on the query complexity makes our results all the more meaningful. Likewise, the weakness of the adversary's goal only strengthens the lower bound.

1.3 Related work

Even and Mansour [8] construct a block cipher $PX : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ from a random permutation $P : \{0,1\}^n \to \{0,1\}^n$ by $PX_{k1.k2}(x) = k2 \oplus P(k1 \oplus x)$. Clearly this is a special case of the FX construction, where $\kappa = 0$. While their motivation for looking at PX was quite different from our reasons to investigate FX, our model and methods are, in fact, quite similar. Our main result can be seen as a natural extension of their work.

The modeling of a block cipher by a family of random permutations has its roots in [15].

Ron Rivest invented DESX by May of 1984, but never described the scheme in any conference or journal paper [13]. DESX was implemented within products of RSA Data Security, Inc., and is described in the documentation for these products [14]. DESX has also been described at conferences organized by RSA DSI, including [18].

Encryption methods similar to DESX have been invented independently. Blaze [3] describes a DES mode of operation in which the ith block of plaintext, x_i, is encrypted using 112-bit key $k.k1$ by $E_{k.k1}(x_i) = s_i \oplus \mathrm{DES}_k(s_i \oplus x)$, where $s_1 s_2 \cdots$ is a stream of bits generated from $k1$ by, say, $s_i = \mathrm{DES}^{(i)}_{k1}(0^{64})$. Here $\mathrm{DES}^{(i)}$ denotes the i-th iterate of DES.

Many authors have suggested methods to increase the strength of DES by changing its internal structure. Biham and Biryukov [1] give ways to modify DES to use key-dependent S-boxes. Their suggestions improve the cipher's strength against differential, linear, and improved Davies' attacks, as well as exhaustive key search. Ciphers constructed using their ideas can exploit existing hardware exactly in those cases where the hardware allows the user to substitute his own S-boxes in place of the standard ones.

1.4 Discussion

UNDERSTANDING OUR RESULT. It may be hard to understand the ramifications of our main theorem, thinking it means more or less than it does. DES, of course, is not a family of random permutations, and we can *not* conclude from our theorem that there does not exist a reasonable machine M which breaks DESX in say, 2^{60} steps, given just a handful of ⟨plaintext, ciphertext⟩ pairs. What we can say is that such a machine would have to exploit structural properties of DES; it couldn't get away with treating DES as a black-box transformation. This contrasts with the sort of machines which have been suggested in the past for doing brute-force attack: they *do* treat the underlying cipher as a black-box transformation.

We note that while remarkable theoretical progress has been made on the linear and differential cryptanalysis of DES (see [2, 12]), thus far these attacks require an impractically large number of plaintext-ciphertext pairs. To date, the only published practical attacks against DES remain of

the key-search variety. The DESX construction was not intended to improve the strength of DES against differential or linear attack, or any other attack which exploits structural properties of DES, and our theorem does not say anything about its resistance to these attacks.

ON EXPORT CONTROLS TIED TO KEY LENGTH. Our results indicate how algorithmically trivial it can be to obtain extra bits of strength against exhaustive key-search attacks. The impact of these extra bits can be especially dramatic when the key length of the block cipher had been intentionally made short.

Consider a block cipher F with a 40-bit key and a 64-bit plaintext. (Some products using such block ciphers have been granted U.S. export approval.) With these parameters, our results guarantee an effective key length (with respect to exhaustive key search) of at least $40+64-1-\lg m = 103-\lg m$ bits. Under the reasonable assumption that $m < 2^{30}$, say, the 40-bit block cipher has been modified, with two XORs, to a new block cipher which needs at least 2^{73}-time for key exhaustive key search.

Allowing weak cryptography to be exported and strong cryptography not to be is a policy which can only make sense when it is impractical, for the given system, to replace the weak mechanism by a strong one. Our results indicate that this impracticality must cover algorithmic changes that are particularly trivial.

1.5 Outline of the paper

In Section 2 we define some basic notation and define what comprises a successful attack in our model. In Section 3 we state and prove our main theorem on the security of the DESX construction. Section 4 is a discussion. Section 5 demonstrates that the analysis underlying our main result is tight. In Section 6 we give some conclusions and open questions.

2 Preliminaries

Let $\mathcal{P}_n$ denote the space of all $(2^n)!$ permutations on n-bits.

We say that $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher if for every $k \in \{0,1\}^\kappa$, $F(k,\cdot) \in \mathcal{P}_n$. We define F_k by $F_k(x) = F(k,x)$. Let $\mathcal{B}_{\kappa,n}$ denote the space of all block ciphers with parameters κ and n as above.

Given $F \in \mathcal{B}_{\kappa,n}$, we define the block cipher $F^{-1} \in \mathcal{B}_{\kappa,n}$ by $F^{-1}(k,y) = F_k^{-1}(y)$ for $k \in \{0,1\}^\kappa$. We interchangeably write $F_k^{-1}(y)$ and $F^{-1}(k,y)$.

Given $F \in \mathcal{B}_{\kappa,n}$, we define the block cipher $FX \in \mathcal{B}_{\kappa+2n,n}$ by $FX(K,x) = k2 \oplus F_k(k1 \oplus x)$, where $K = k.k1.k2$, $|k| = \kappa$ and $|k1| = |k2| = n$. We interchangeably write $FX_K(x)$ and $FX(K,x)$.

Given a partially defined function F from a subset of $\{0,1\}^m$ to a subset of $\{0,1\}^n$ we denote the domain and range of F by $\mathrm{Dom}(F)$ and $\mathrm{Range}(F)$, and define $\overline{\mathrm{Dom}}(F) = \{0,1\}^m - \mathrm{Dom}(F)$ and $\overline{\mathrm{Range}}(F) = \{0,1\}^n - \mathrm{Range}(F)$.

We denote by $x \xleftarrow{R} S$ the act of choosing x uniformly from S. We denote by $\Pr[A_1;\ A_2;\ \ldots : E]$ the probability of event E after performing actions $A_1, A_2, \ldots$.

Definition 2.1 *A generic key-search adversary is an algorithm A with access to three oracles, E, F and F^{-1}. Thus, A may make queries of the form $E(P)$, $F_k(x)$ or $F_k^{-1}(y)$. An (m,t) generic key-search adversary is a key-search adversary that makes m queries to the E oracle and a total of t queries to the F and F^{-1} oracles.*

For brevity, we will sometimes drop "generic" from our terminology. Note that A supplies the value of k as part of its queries to the F and F^{-1} oracles. We denote by $A^{E,\ F,\ F^{-1}}$ the adversary A interacting with oracles E, F and F^{-1}.

We now define what it means for a generic key-search adversary A to have an attack of a certain specified effectiveness. We begin by choosing a random block cipher F having κ-bit keys and n-bit blocks. This means that we select a random permutation $F_k \xleftarrow{R} \mathcal{P}_n$ for each κ-bit key k. Thus each F_k is chosen independently of each $F_{k'}$, for $k \neq k'$. Then we give A three oracles, E, F and F^{-1}. The F and F^{-1} oracles compute their respective functions. The encryptions oracle E, on input x, either computes $FX_K(x)$ for a random $(\kappa+2n)$-bit key K or computes $\pi(x)$, for a random permutation $\pi \xleftarrow{R} \mathcal{P}_n$. The adversary's job is to guess which type of encryption oracle she has. Our convention is that A outputs a 1 to guess that the encryption oracle is computing $FX_K(x)$. The adversary's *advantage* is her probability of guessing right, normalized to a $[-1,1]$ scale: -1 indicates a strategy that always guesses wrong; 1 indicates a strategy that always guesses correctly; guessing at random, or always guessing the same way, will give an advantage of 0.

Definition 2.2 *Let $\kappa, n \geq 0$ be integers, and let $\epsilon \geq 0$ be a real number. Generic key-search adversary A is said to* ϵ-break *the FX-scheme with parameters κ, n if*

$$\begin{aligned} \mathrm{Adv}_A &\stackrel{\text{def}}{=} \Pr\left[F \xleftarrow{R} \mathcal{B}_{\kappa,n};\ K \xleftarrow{R} \{0,1\}^{\kappa+2n} :\ A^{FX_K,\ F,\ F^{-1}} = 1\right] - \\ &\qquad \Pr\left[F \xleftarrow{R} \mathcal{B}_{\kappa,n};\ \pi \xleftarrow{R} \mathcal{P}_n :\ A^{\pi,\ F,\ F^{-1}} = 1\right] \\ &\geq \epsilon\,. \end{aligned}$$

The above definition uses a very liberal notion of adversarial success. We are not demanding that, say, A recover K; nor do we ask A to decrypt a random $FX_K(x)$ or to produce a not-yet-asked $\langle x, FX_K(x)\rangle$ pair. Instead, we only ask A to make a good guess as to whether the ⟨plaintext, ciphertext⟩ pairs she has been receiving really *are* FX-encryptions, as opposed to random nonsense unrelated to F. The liberal notion of success is chosen to make our main result stronger: an adversary's inability to succeed becomes all the more meaningful.

3 Security of the DESX Construction

We now prove a bound on the security of FX against generic key-search attacks.

Theorem 3.1 *Let A be an (m,t) generic key-search adversary that ϵ-breaks the FX-scheme with parameters κ, n. Then $\epsilon \leq mt \cdot 2^{-\kappa-n+1}$.*

Proof: Before going into the detailed formal proof, we first give some intuition for why the proof works. Clearly, the FX construction is highly nonrandom if one makes all possible queries to the E, F and F^{-1} oracles. However, to defeat the adversary it suffices if the answers to its relatively few queries are random. For intuition, we erroneously think of F as a family of random functions (the formal analysis takes into account the fact that F_k is a permutation). We conceptually view F as undefined; as queries from the adversary come in we choose values of $F_k(x)$ at random. Note that queries to $E(x)$ implicitly make queries to F. If in computing $E(x)$ we make a "fresh" query to F (one that hasn't been made before), we generate a fresh answer that is random and independent of the entire history of the attack. This fresh randomness ensures that the resulting value of $E(x)$ will be random. However, randomness cannot be guaranteed when new queries depend on previously determined values. We show that if the adversary doesn't make many queries, then these bad events happen with low probability.

By a standard argument we may assume that A is deterministic (note that A may be computationally unbounded).[2] We may also assume that A always asks exactly m queries of her first oracle, which we shall call her E-oracle. (In the experiment that defines A's advantage, E was instantiated by either an FX_K-oracle or a π-oracle.) We may assume that A always asks exactly t queries (total) to her second and third oracles, which we shall call her F- and F^{-1}- oracles. We may further assume that A never repeats a query to an oracle. We may assume that if $F(k, x)$ returns an answer y, then there is no query (neither earlier nor later) of $F^{-1}(k, y)$. All of the above assumptions are without loss of generality in the sense that it is easy to construct a new adversary, A', that obeys the above constraints and has the same advantage as A.

We begin by considering two different games that adversary A might play. This amounts to specifying how to simulate a triple of oracles, $\langle E, F, F^{-1}\rangle$, for the benefit of A.

A FIRST GAME. The first game we consider, Game R (for "random"), will exactly correspond to the experiment which defines the second addend in the expression for the advantage:

$$P_R = \Pr\left[A^{\pi,\ F,\ F^{-1}} = 1\right].$$

The definition of Game R will be defined to contain several extra (and seemingly irrelevant) steps. These steps aren't needed in order to behave in a manner which is identical (as far as A sees) to the manner of behavior defining P_R; these steps are used, instead, to facilitate our analysis. To identify these "irrelevant" instructions we put them in italics. Game R is defined in Figure 1.

Let $\Pr_R[\cdot]$ denote the probability of the specified event with respect to Game R. From the definition of Game R we can see that:

Claim 3.1 $\Pr_R\left[A^{E,F,F^{-1}} = 1\right] = P_R$.

A SECOND GAME. Now we define a second game, Game X. It will exactly correspond to the experiment which defines the first term in the expression for the advantage:

$$P_X = \Pr\left[A^{FX_K,\ F,\ F^{-1}} = 1\right].$$

[2] Roughly, given unlimited computational capabilities, A can derandomize its strategy by exhaustively searching through its possible random choices, computing the effectiveness of the resulting attack, and then choosing the most efficacious choice.

Game R

Initially, let F and E be undefined. *Flag* ***bad*** *is initially unset. Randomly choose* $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$. Then answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. Choose $C \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(E)$.
2. *If* $F_{k^*}(P \oplus k_1^*)$ *is defined, then set* ***bad***.
   If $F_{k^*}^{-1}(C \oplus k_2^*)$ *is defined, then set* ***bad***.
3. Define $E(P) = C$ and return C.

$\boxed{F}$ On oracle query $F_k(x)$:

1. Choose $y \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(F_k)$.
2. *If* $k = k^*$ *and* $E(x \oplus k_1^*)$ *is defined then set* ***bad***.
   If $k = k^*$ *and* $E^{-1}(y \oplus k_2^*)$ *is defined then set* ***bad***.
3. Define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. Choose $x \in \{0,1\}^n$ uniformly from $\overline{\text{Dom}}(F_k)$.
2. *If* $k = k^*$ *and* $E^{-1}(y \oplus k_2^*)$ *is defined then set* ***bad***.
   If $k = k^*$ *and* $E(x \oplus k_1^*)$ *is defined then set* ***bad***.
3. Define $F_k(x) = y$ and return x.

Game X

Initially, let F and E be undefined. *Flag* ***bad*** *is initially unset.* Randomly choose $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$. Then answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. Choose $C \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(E)$.
2. If $F_{k^*}(P \oplus k_1^*)$ is defined, then $C \leftarrow F_{k^*}(P \oplus k_1^*) \oplus k_2^*$ *and set* ***bad***.
   Else if $F_{k^*}^{-1}(C \oplus k_2^*)$ is defined, then *set* ***bad*** *and* goto Step 1.
3. Define $E(P) = C$ and return C.

$\boxed{F}$ On oracle query $F_k(x)$:

1. Choose $y \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(F_k)$.
2. If $k = k^*$ and $E(x \oplus k_1^*)$ is defined then $y \leftarrow E(x \oplus k_1^*) \oplus k_2^*$ *and set* ***bad***.
   Else If $k = k^*$ and $E^{-1}(y \oplus k_2^*)$ is defined then *set* ***bad*** *and* goto Step 1.
3. Define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. Choose $x \in \{0,1\}^n$ uniformly from $\overline{\text{Dom}}(F_k)$.
2. If $k = k^*$ and $E^{-1}(y \oplus k_2^*)$ is defined then $x \leftarrow E^{-1}(y \oplus k_2^*) \oplus k_1^*$ *and set* ***bad***.
   Else if $k = k^*$ and $E(x \oplus k_1^*)$ is defined then *set* ***bad*** *and* goto Step 1.
3. Define $F_k(x) = y$ and return x.

Figure 1: Games R and X.

Once again, the definition of Game X will be defined to contain some "irrelevant" instructions, which, for clarity, are indicated in italics. Game X is defined in Figure 1.

The intuition behind Game X is as follows. We *try* to behave like Game R, choosing a random (not-yet-provided) answer for each $E(P)$, and a random (not-yet-provided for this k) answer for each $F_k(x)$, $F_k^{-1}(y)$. Usually this works fine for getting behavior which looks like the experiment defining P_X. But sometimes it doesn't work, because an "inconsistency" would be created between the FX-answers and the F/F^{-1}-answers. Game X is vigilant in checking if any such inconsistencies are being created. If it finds an inconsistency about to be created, it *changes* the value which it had "wanted" to answer in order to *force* consistency. Whenever Game X resorts to doing this it sets the flag ***bad***. In the analysis, we "give up" (regard the adversary as having won) any time this happens.

Let $\Pr_X[\cdot]$ denote the probability of the specified event with respect to Game X. The definition of Game X looks somewhat further afield from the experiment which defines P_X. Nonetheless, we claim the following:

Claim 3.2 $\Pr_X\left[A^{E,F,F^{-1}}=1\right]=P_X$.

The proof of this claim is in the appendix.

BOUNDING THE ADVANTAGE BY $\Pr_R[\mathsf{BAD}]$. In either Game R or Game X, let BAD be the event that, at some point in time, the flag ***bad*** gets set. Games R and X have been defined so as to coincide up until event BAD. To see this, note that the corresponding oracles in these games are identical except for, in each case, Step 2. For each pair of oracles, Step 2 executes identical tests and based on the outcome of the test either does nothing in both cases or sets ***bad*** in both cases (and other actions, in which the oracles will differ in their behavior). Thus, any circumstance that causes Game R and Game X to execute different instructions will also cause both games to set ***bad***. The following two claims follow directly from this fact.

Claim 3.3 $\Pr_R[\mathsf{BAD}]=\Pr_X[\mathsf{BAD}]$.
Claim 3.4 $\Pr_R\left[A^{E,F,F^{-1}}=1|\overline{\mathsf{BAD}}\right]=\Pr_X\left[A^{E,F,F^{-1}}=1|\overline{\mathsf{BAD}}\right]$.

What we have shown so far allows us to bound the adversary's advantage by $\Pr_R[\mathsf{BAD}]$.

Claim 3.5 $\mathrm{Adv}_A\leq\Pr_R[\mathsf{BAD}]$.

The argument is quite simple:

$$\begin{aligned}
\mathrm{Adv}_A &= P_X-P_R\\
&= \Pr_X\left[A^{E,F,F^{-1}}=1\right]-\Pr_R\left[A^{E,F,F^{-1}}=1\right] && \textit{(Claims 3.1, 3.2)}\\
&= \Pr_X[A=1|\overline{\mathsf{BAD}}]\Pr_X[\overline{\mathsf{BAD}}]+\Pr_X[A=1|\mathsf{BAD}]\Pr_X[\mathsf{BAD}]-\\
&\quad \Pr_R[A=1|\overline{\mathsf{BAD}}]\Pr_R[\overline{\mathsf{BAD}}]-\Pr_R[A=1|\mathsf{BAD}]\Pr_R[\mathsf{BAD}]\\
&= \Pr_R[\mathsf{BAD}](\Pr_X[A=1|\mathsf{BAD}]-\Pr_R[A=1|\mathsf{BAD}]) && \textit{(Claims 3.3, 3.4)}\\
&\leq \Pr_R[\mathsf{BAD}]
\end{aligned}$$

Initially, let F and E be undefined. Answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. Choose C uniformly from $\overline{\text{Range}}(E)$.
2. Define $E(P) = C$ and return C.

$\boxed{F}$ On oracle query $F_k(x)$:

1. Choose y uniformly from $\overline{\text{Range}}(F_k)$
2. Define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. Choose x uniformly from $\overline{\text{Dom}}(F_k)$.
2. Define $F_k(x) = y$ and return x.

After all the queries have been answered:
Flag ***bad*** *is initially unset.*
Randomly choose $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$.
If $\exists\ x$ *such that* $F_{k^*}(x)$ *and* $E(x \oplus k_1^*)$ *are both defined then set* ***bad****.*
If $\exists\ y$ *such that* $F_{k^*}^{-1}(y)$ *and* $E^{-1}(y \oplus k_2^*)$ *are both defined then set* ***bad****.*

Figure 2: Game R'

A THIRD GAME. We have reduced our analysis to bounding $\Pr_R[\mathsf{BAD}]$. To bound $\Pr_R[\mathsf{BAD}]$, let us imagine playing Game R a little bit differently. Instead of choosing k^*, k_1^*, k_2^* at the beginning, we choose them at the end. Then we set ***bad*** to be *true* or *false* depending on whether or not the choice of k^*, k_1^*, k_2^* we've just made would have caused ***bad*** to be set to true in Game R (where the choice was made at the beginning). The new game, Game R', is described in Figure 2. From the definition of Game R' we see that:

Claim 3.6 $\Pr_R[\mathsf{BAD}] = \Pr_{R'}[\mathsf{BAD}]$.

COMPLETING THE PROOF. Now that we have sufficiently manipulated the games a simple calculation suffices to bound $\Pr_{R'}[\mathsf{BAD}]$, and, thereby, to bound Adv_A.

After having run the body of Game R', not having yet chosen k^*, k_1^*, k_2^*, let us simply count how many of the $2^{\kappa+2n}$ choices for (k^*, k_1^*, k_2^*) will result in ***bad*** getting set.

Fix any possible values for E and F which can arise in Game R'. Let $|E|$ denote the number of defined values $E(P)$, and let $|F|$ denote the number of defined values $F_k(x)$. Note that $|E| = m$ and $|F| = t$. Fix E and F. Call (k^*, k_1^*, k_2^*) *collision-inducing* (with respect to E and F) if there is some defined $y = F_k(x)$ and some defined $C = E(P)$ such that

$$k^* = k \quad \text{and} \quad (P \oplus k_1^* = x \quad \text{or} \quad C \oplus k_2^* = y).$$

Every choice of (k^*, k_1^*, k_2^*) which results in setting ***bad*** is collision-inducing, so it suffices to upper bound the number of collision-inducing (k^*, k_1^*, k_2^*).

Claim 3.7 *Fix E, F, where $|E| = m$ and $|F| = t$. There are at most $2mt \cdot 2^n$ collision-inducing $(k^*, k_1^*, k_2^*) \in \{0,1\}^\kappa \times \{0,1\}^n \times \{0,1\}^n$.*

The reason is as follows: for each defined $(P, E(P))$, $(k, x, F_k(x))$ there are at most $2 \cdot 2^n$ points (k^*, k_1^*, k_2^*) which induce a collision between these two points: they are the points $(k^*, k_1^*, k_2^*) \in \{k\} \times \{x \oplus P\} \times \{0,1\}^n\} \cup \{k\} \times \{0,1\}^n \times \{y \oplus C\}\}$. Now there are only mt pairs of such points, so the total number of collision-inducing (k^*, k_1^*, k_2^*) is as claimed.

Finally, in Game R' we choose a triple (k^*, k_1^*, k_2^*) at random, independent of E and F, so the chance that the selected triple is collision-inducing (for whatever E and F have been selected) is at most $2mt \cdot 2^n / 2^{\kappa+2n} = mt \cdot 2^{-\kappa-n+1}$. Pulling everything together, this probability bounds Adv_A, and we are done. $\diamondsuit$

4 Discussion

HEALTH WARNINGS. We emphasize that when F is a concrete block cipher, not a random one, its internal structure can interact with the FX-construction in such a way as to obviate the construction's benefits. As a trivial example, if F *already* has the structure that it XORs plaintext and ciphertext with key material, then doing it *again* is certainly of no utility.

Our model considers how much FX_K looks like a random permutation (when key K is random and unknown). It should be emphasized that some constructions which use block ciphers—particularly hash function constructions—assume something more of the underlying block cipher. The current results imply nothing about the suitability of FX in constructions which are *not* based on FX_K resembling a random permutation when K is random and unknown.

We also note that our analysis as stated only considers chosen-plaintext attacks and does not establish resistance to chosen-ciphertext attacks. However, it is straightforward to adapt our techniques to analyze chosen-ciphertext attacks, as was done in [8]. To do this, provide A an oracle for FX^{-1}, in addition to her other oracles. Now m will count the sum of the number of queries to the FX and FX^{-1} oracles. Theorem 3.1 will then continue to hold. The proof changes very little.

STRUCTURE IN THE BLOCK CIPHER F WHEN F = DES. There is one structural property of DES which has been suggested to assist in brute-force attacks: the DES key-complementation property. This property comprises a significant sense in which DES is not behaving like a family of (independent) random permutations. To "factor out" the key-complementation property just think of DES as having a single key bit fixed. Then one can conclude that if this is the only structural property of DES to be exploited by a generic key-search attack, DESX will still limit the attack's advantage to $tm \cdot 2^{-55-64+1} = tm \cdot 2^{-118}$.

SETTING $k1 = k2$. As mentioned in the introduction, the simpler constructions $FX^{\mathrm{pre}}_{k.k1}(x) = F_k(x \oplus k1)$ and $FX^{\mathrm{post}}_{k.k1}(x) = k1 \oplus F_k(x)$ don't significantly improve F's strength against generic

key search attacks. But what about

$$FX'_{k.k1}(x) = k1 \oplus F_k(x \oplus k1)?$$

Is it OK to use the same key inside and out? In fact this does work, in the sense that Theorem 3.1 still goes through, the proof little changed. We analyzed the more "standard" general construction, with two keys, but the more restricted choice has the advantage of a smaller key-size, with no obvious loss of security.

NICER KEY LENGTHS. A minor inconvenience of DESX is its strange key size. In applications it would sometimes be preferable to extend the definition of DESX to use arbitrary-length keys, or else to use keys of some fixed but more convenient length. Standard key-separation techniques can be used.

We give one extension of DESX to arbitrary-length keys, as follows. Let $X_{1...\ell}$ denotes the first ℓ bits of X, let SHA-1 be the map of the NIST Secure Hash Standard, and let C, $C1$ and $C2$ be fixed, distinct, equal-length strings. When $|K| \neq 184$, we can define $\mathrm{DESX}_K(x)$ to be equal to $\mathrm{DESX}_{K'}(x)$ where K' is defined as follows:

- If $|K| = 56$ then $K' = K.0^{64}, 0^{64}$.
- Otherwise, $K' = k.k1.k2$, where $\begin{cases} k &= \text{SHA-1}(C.K)_{1...56}, \\ k1 &= \text{SHA-1}(C1.K)_{1...64}, \text{ and} \\ k2 &= \text{SHA-1}(C2.K)_{1...64} \end{cases}$

Note that when $|K| = 56$, $\mathrm{DESX}_K(x) = \mathrm{DES}_K(x)$.

DIFFERENTIAL AND LINEAR CRYPTANALYSIS. OPERATIONS BESIDES XOR. We emphasize that the DESX construction was never intended to add strength against differential or linear cryptanalysis. The attacks of [2, 12] do not represent a threat against DES when the cipher is prudently employed (e.g., when a re-key is forced before an inordinate amount of text has been acted on); until these attacks are improved, it suffices that the DESX construction does not render differential or linear attack any *easier*.

Nonetheless, the proof of Theorem 3.1 goes through when $\oplus$ is replaced by a variety of other operations, and some of these alternatives may help to defeat attacks which were not addressed by our model, including differential and linear cryptanalysis. In particular, an attractive alternative to DESX may be the construction $\mathrm{DESP}_{k.k1.k2}(x) = k2 + \mathrm{DES}_k(k1 + x)$, where $LR + L'R' \stackrel{\text{def}}{=} L\hat{+}L' \,.\, R\hat{+}R'$, where $|L| = |R| = |L'| = |R'| = 32$ and $\hat{+}$ denotes addition modulo 2^{32}. Burt Kaliski has suggested such alternatives, and analyzed their security with respect to differential and linear attacks [9].

5 Our Bound is Tight

We have shown that the adversary's advantage is at most $t \cdot 2^{-n-\kappa+1+\lg m}$. Turning this around, the adversary needs $\epsilon 2^{n+\kappa-1-\lg m}$ queries to the F/F^{-1} oracles to achieve an ϵ-advantage. We now show that for a wide range of m (comprising all m that would be considered in practice), an

attacker can achieve an ϵ-advantage using very close to $2^{n+\kappa+4-\lg m}$ queries to the F/F^{-1} oracles (the exact bound is given in Corollary 5.2). This follows as a corollary of a more ambitious attack. This attack recovers a key $K' = k'.k1'.k_2'$ that is consistent with the encryptions under FX of m plaintexts chosen before F/F^{-1} oracles queries are made.

Theorem 5.1 *Let m be even, $m < 2^n$ and $\epsilon < \frac{1}{2}$. Let block cipher F be uniformly distributed over $\mathcal{B}_{\kappa,n}$ and let key K be uniformly distributed over $\{0,1\}^{\kappa+2n}$. Then there exists an adversary $A(m,\epsilon)$ that initially makes m distinct queries $t_1,\ldots,t_m$ (the test set) to an oracle computing FX_K. Adversary A then makes*

$$\left(2^{n+\kappa+1-\lg m} + 2^n + 2^\kappa\right)\left(\epsilon + \epsilon^2\right)$$

expected queries to the F/F^{-1} oracles. With probability at least ϵ it returns a K' such that $FX'_K(t_i) = FX_K(t_i)$ for $1 \le i \le m$. The probability is taken over the choice of F, K and A's coin tosses.

It follows that our analysis is essentially tight, given our measure on the attacker's resources, which roughly corresponds to time. We note that in practice it is also important to consider the memory requirements of an attack. Conceivably, there exists stronger attacks than require the same amount of time but much less memory. However, if the time requirements are sufficiently high, the memory issue becomes moot. However, it is an interesting open question whether imposing a reasonable space bound can allow us to improve our time bound.

For reasonable values of m, the task performed by $A(m,\epsilon)$ is at least as strong as simply distinguishing FX from a purely random permutation. To see this, consider any family of permutations $\{FX_K\}$ on $\{0,1\}^n$, where $|K| = \kappa+2n$. We say that π is *plausible* if for some K, $\pi(x_i) = FX_K(x_i)$ for $1 \le i \le m$. If π is chosen at random, then by a simple counting argument the probability that it is plausible is at most

$$\rho(\kappa,n,m) \stackrel{\text{def}}{=} \frac{2^{\kappa+2n}}{2^n(2^n-1)\cdots(2^n-m+1)}\,.$$

For example, if $\kappa \le n$, $n > 20$ and $m > 6$ then $\rho < 10^{-24}$. So an attacker who outputs a 1 iff she finds a consistent K has an advantage of $\epsilon - \rho(\kappa,n,m)$, which is essentially ϵ.

A minor technical point is that our lower bound considered attackers with worst-case instead of expected case bounds. However, we can convert the expectation into a worst case bound by observing that if an expected value is at most Q then with probability $\frac{1}{2}$ it is at most $2Q$. Hence, for $\epsilon < \frac{1}{2}$ we can set the attacker A in Theorem 5.1 to find a consistent K with probability 2ϵ, and time out if A takes more than twice its expected number of F/F^{-1} queries. The resulting attack uses at most

$$\begin{aligned} t &\le \left(2^{n+\kappa+2-\lg m} + 2^{n+1} + 2^{\kappa+1}\right)\left(2\epsilon + 4\epsilon^2\right) \\ &\le \left(2^{n+\kappa+4-\lg m} + 2^{n+3} + 2^{\kappa+3}\right)\epsilon \end{aligned}$$

worst-case queries to the F/F^{-1} oracles.

Finally, since the advantage is $\epsilon - \rho(\kappa,n,m)$ we can set ϵ to be $\rho(\kappa,n,m)$ bigger than the desired advantage, giving the following corollary.

Corollary 5.2 *Let m be even, $m < 2^n$ and $\epsilon < \frac{1}{2} - \rho(\kappa, n, m)$. Let block cipher F be uniformly distributed over $\mathcal{B}_{\kappa,n}$, key K be uniformly distributed over $\{0,1\}^{\kappa+2n}$ and permutation π be a uniformly distributed over $\mathcal{P}_n$. There exists an attacker $A(m, \epsilon)$ that makes m queries to oracle E (computing either FX_K or π) and makes*

$$\left(2^{n+\kappa+4-\lg m} + 2^{n+3} + 2^{\kappa+3}\right)(\epsilon + \rho(\kappa, n, m))$$

queries to the F/F^{-1} oracles. A solves the FX-or-π game with advantage at least ϵ.

The rest of this section is devoted to the proof of Theorem 5.1.

To motivate our attack, we can view the FX block cipher as choosing a random key k and then applying the Even-Mansour construction to the function F_k. We can therefore trivially adapt Daemen's chosen-plaintext attack [5] on the Even-Mansour construction [8]. Unfortunately, we don't know the value of k, so we instead try all possible ones. For completeness, we describe the attack and calculate the amount of work required to have probability ϵ of recovering the key.

5.1 Preliminaries

Assume that m is even, $m \leq 2^n$, and $\epsilon < \frac{1}{2}$. Fix a constant $C \in \{0,1\}^n - \{0^n\}$. For any function G, define $G^{\Delta}(x) = G(x \oplus C) \oplus G(x)$. Given an oracle for G one can compute G^{Δ} by making two calls. Let the secret key $K = k.k1.k2$. Let E by a synonym for FX. By our definitions and simple algebra we have

$$E_K^{\Delta}(x) = F_k^{\Delta}(x \oplus k1) = F_k^{\Delta}(x \oplus C \oplus k1).$$

5.2 The key-search attack

The attacker A works as follows. A uses oracles computing FX_K (for the correct $K = k.k1.k2$) and F. Attacker A takes as parameters m, the maximum number of queries it is allowed to make to the FX_K oracle and ϵ a required lower bound on its probability of producing a key K' that gives consistent results on the m queries it made to FX_K.

$A(m, \epsilon)$

1. Choose $x_1, \ldots, x_{m/2} \in \{0,1\}^n$ arbitrarily so that

   $$\text{TEST} = x_1, \ldots, x_{m/2}, x_1 \oplus C, \ldots, x_{m/2} \oplus C$$

   has m distinct elements.

2. Using the FX_K oracle, compute $FX_K(t)$ for $t \in \text{TEST}$, and then compute

   $$FX_K^{\Delta}(x_1), \ldots, FX_K^{\Delta}(x_{m/2}).$$

3. For i from 1 to $\ell = \left\lceil \frac{-2^n \ln(1-\epsilon)}{m} \right\rceil$ do
   Choose $r \xleftarrow{R} \{0,1\}^n$

For all $k' \in \{0,1\}^\kappa$, $1 \le j \le m/2$ do
  If $F^{\Delta}_{k'}(r) = FX^{\Delta}_K(x_j)$
    /* *Hope that* $k' = k$ *and* r *is either* $x_j \oplus k1$ *or* $x_j \oplus C \oplus k1$ */
    For $k1' \in \{x_j \oplus r, x_j \oplus C \oplus r\}$
      $k2' = F_{k'}(x_1 \oplus k1') \oplus FX_K(x_1)$; K'=k'.k1'.k2'
      If $FX_{K'}(t) = FX_K(t)$ for $t \in$ TEST
        Return K'

5.3 Analysis of the attack

To analyze this attack we first bound the oracle-query complexity of testing each r. We then compute how many r's are needed in order to succeed with probability ϵ.

We say that r is *good* if it is equal to $x_j \oplus k1$ or $x_j \oplus C \oplus k1$ for some j. If r is good then as soon as the attacker tries $k' = k$ (remember she tries them all) she will obtain the correct values for $k1'$ and then $k2'$ (though she may try some incorrect values as well).

We now bound the expected cost of trying each r. For each value of r (good or bad), the attacker must go through, in the worst case, all 2^κ values for k'. For each value of k', it makes 2 calls to the F oracle in order to compute $F^{\Delta}_{k'}(r)$, giving a base cost of $2^{\kappa+1}$ calls to the F oracle. Given a promising (j, k'), where $F^{\Delta}_{k'}(r) = FX^{\Delta}_K(x_j)$, the attacker generates 2 guesses $k1'$, and for each $k1'$ she makes an additional call to the F oracle to compute $k2'$. Testing $k', k1'$ and $k2'$ requires no further oracle calls.

We note that for any r, when $k' \neq k$ the distribution on $F^{\Delta}_{k'}(r)$ is random even conditioned on the answers to all of the FX oracle queries. Thus, the expected number of j such that (k', j) is promising is at most $m/2^{n+1}$. When $k' = k$, then in the worst case, $m/2$ promising values of (k', j) are tested. Therefore, the expected extra number of oracle queries needed to evaluate promising candidates is at most $m + m2^\kappa/2^n$ for each value of r selected. Thus, for each random r selected, a total of at most

$$2^{\kappa+1} + m + m2^{\kappa-n}$$

expected queries are required.

It remains to bound the number of r's that must be tried in order to select a good r with probability at least ϵ. There are exactly m good r out of 2^n possibilities. Thus, the probability that ℓ randomly selected values for r will fail to be good is at most $(1 - m/2^n)^\ell$. We thus need to select ℓ so that $(1 - m/2^n)^\ell \le 1 - \epsilon$. Using the identity $(1 + a)^b \le e^{ab}$, it suffices to achieve

$$e^{-m\ell/2^n} \le 1 - \epsilon,$$

or equivalently,

$$\ell \ge \frac{-2^n \ln(1-\epsilon)}{m}.$$

For $0 < \epsilon < \frac{1}{2}$, $-\ln(1-\epsilon) < \epsilon + \epsilon^2$, so it suffices that

$$\ell \ge \frac{2^n(\epsilon + \epsilon^2)}{m}.$$

Summarizing the above, there is an attack which finds a consistent key $K' = k'.k1'.k2'$ with probability ϵ using m queries to the FX_K oracle, and at most expected

$$\left(2^{n+\kappa+1-\lg m} + 2^n + 2^\kappa\right)\left(\epsilon + \epsilon^2\right)$$

queries to the F/F^{-1} oracles. The theorem follows. $\diamondsuit$

6 Open Problems and Conclusions

ANALYSIS OF OTHER MULTIPLE ENCRYPTION SCHEMES. The model we have used to upper bound the worth of key search applies to many other block-cipher based constructions. For example, it would be interesting to apply this model to bound the maximal advantage an adversary can get for triple DES with three distinct keys, or triple DES with the first and third keys equal, or the method of [4]. It would be interesting to demonstrate that some construction has a better effective key length than DESX (e.g., $k + n - 1$ bits).

USE IT! Work within some standards bodies continues to specify encryption based on DES in its most customary mode of operation. We recommend DESX (or one of its variants, as in Section 4). DESX is efficient, DES-compatible, patent-unencumbered, and resists generic key-search attacks. In virtually every way, DESX would seem to be a better DES than DES.

Acknowledgments

Mihir Bellare was closely involved in the early stages of our investigation. Burt Kaliski, Ron Rivest, and anonymous referees provided useful comments and information.

References

[1] E. BIHAM AND A. BIRYUKOV, "How to strengthen DES using existing hardware." *Advances in Cryptology— ASIACRYPT '94.* Springer-Verlag (1994).

[2] E. BIHAM AND A. SHAMIR, *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag (1993).

[3] M. BLAZE, "A cryptographic file system for UNIX." *1st ACM Conference on Computer and Communications Security,* 9–16 (November 1993).

[4] D. COPPERSMITH, D. JOHNSON AND M. MATYAS, "A proposed mode for triple-DES encryption." IBM Journal of Research and Development, vol. 40, no. 2, 253–261 (1996).

[5] J. DAEMEN, "Limitations of the Even-Mansour construction" (abstract of a rump-session talk). *Advances in Cryptology— ASIACRYPT '91.* Lecture Notes in Computer Science, vol. 739, 495–498, Springer-Verlag (1992).

[6] W. DIFFIE AND M. HELLMAN, "Exhaustive cryptanalysis of the NBS Data Encryption Standard." *Computer*, vol. 10, no. 6, 74–84 (June 1977).

[7] ELECTRONIC FRONTIER FOUNDATION, *Cracking DES: Secrets of Encryption Research, Wiretap Politics, & Chip Design.* O'Reilly and Associates (1998).

[8] S. EVEN AND Y. MANSOUR, "A construction of a cipher from a single pseudorandom permutation." *Journal of Cryptology*, vol. 10, no. 3, 151–162 (Summer 1997). Earlier version in *Advances in Cryptology— ASIACRYPT '91.* Lecture Notes in Computer Science, vol. 739, 210–224, Springer-Verlag (1992).

[9] B. KALISKI, *personal communication* (April 1996).

[10] B. KALISKI AND M. ROBSHAW, "Multiple encryption: weighing security and performance," *Dr. Dobb's Journal*, 123–127 (January 1996).

[11] J. KILIAN AND P. ROGAWAY, "How to protect DES against exhaustive key search." *Advances in Cryptology— CRYPTO '96.* Lecture Notes in Computer Science, vol. 1109, 252–267, Springer-Verlag (1996). Earlier version of this paper.

[12] M. MATSUI, "The first experimental cryptanalysis of the data encryption standard." *Advances in Cryptology— CRYPTO '94.* Lecture Notes in Computer Science, vol. 839, 1–11, Springer-Verlag (1994).

[13] R. RIVEST, *personal communication* (1995, 1996).

[14] RSA Data Security, Inc.. Product documentation, "Mailsafe Note #3."

[15] C. SHANNON, "Communication theory of secrecy systems." Bell Systems Technical Journal, 28(4), 656–715 (1949).

[16] P. VAN OORSCHOT AND M. WIENER, "Parallel collision search with cryptanalytic applications." *Journal of Cryptology*, vol. 12, no. 1, 1–28 (1999) Earlier version in *2nd ACM Conference on Computer and Communications Security*, 210–218 (1994).

[17] M. WIENER, "Efficient DES key search." Technical Report TR-244, School of Computer Science, Carleton University (May 1994). Reprinted in *Practical Cryptography for Data Internetworks*, W. Stallings, editor, IEEE Computer Society Press, 31–79 (1996).

[18] Y. YIN, The 1995 RSA Laboratories Seminar Series, "Future directions for block ciphers." Seminar proceedings (page 23) for a talk given in Redwood Shores, California (August 1995).

A Proof of Claim 3.2

We first define a new game, denoted Game X', which matches more directly the definition of the experiment defining P_X. Game X' is defined in Figure 3.

First, note that no adversary can distinguish between playing Game X' and playing with oracles $\langle FX_K, F, F^{-1} \rangle$ drawn according to the experiment defining P_X. Indeed the only difference

Initially, let F be undefined. Randomly choose $k^* \xleftarrow{R} \{0,1\}^\kappa$, $k_1^*, k_2^* \xleftarrow{R} \{0,1\}^n$. Then answer each query the adversary makes as follows:

$\boxed{E}$ On oracle query $E(P)$:

1. If $F_{k^*}(P \oplus k_1^*)$ is defined, return $F_{k^*}(P \oplus k_1^*) \oplus k_2^*$.
2. Otherwise, choose y uniformly from $\overline{\text{Range}}(F_{k^*})$, define $F_{k^*}(P \oplus k_1^*) = y$ and return $y \oplus k_2^*$.

$\boxed{F}$ On oracle query $F_k(x)$:

1. If $F_k(x)$ is defined, return $F_k(x)$.
2. Else, choose $y \in \{0,1\}^n$ uniformly from $\overline{\text{Range}}(F_k)$, define $F_k(x) = y$ and return y.

$\boxed{F^{-1}}$ On oracle query $F_k^{-1}(y)$:

1. If $F_k^{-1}(y)$ is defined, return $F_k^{-1}(y)$.
2. Else, choose $x \in \{0,1\}^n$ uniformly from $\overline{\text{Dom}}(F_k)$, define $F_k(x) = y$ and return x.

Figure 3: Game X'

between these scenarios is that Game X' generates values for E and F by "lazy evaluation," whereas the experiment defining P_X would generate these values all at the beginning. Thus $\Pr_{X'}\left[A^{E,F,F^{-1}} = 1\right] = P_X$.

We want to show that $\Pr_X\left[A^{E,F,F^{-1}} = 1\right] = \Pr_{X'}\left[A^{E,F,F^{-1}} = 1\right]$: no adversary A can distinguish whether she is playing Game X or X'. We emphasize that A's ability to distinguish between Games X and X' is based strictly on the input/output behavior of the oracles; the adversary can not see, for example, whether or not the flag ***bad*** has been set.

We will show something even stronger than that Games X and X' look identical to any adversary. Observe that both Game X and Game X' begin with random choices for k^*, k_1^* and k_2^*. We show that, for any particular values of k^*, k_1^* and k_2^*, Game X with these initial values of k^*, k_1^* and k_2^* is identical, to the adversary, to Game X' with these same initial values of k^*, k_1^* and k_2^*. So, for the remainder of the proof, we consider k^*, k_1^* and k_2^* to have fixed, arbitrary values.

A basic difference between Games X and X' is that Game X separately defines both E and F_{k^*} while Game X' only defines F_{k^*} and computes $E(P)$, in response to a query P, by $F_{k^*}(P \oplus k_1^*) \oplus k_2^*$. The essence of our argument is that Game X can *also* be viewed as answering its $E(P)$ queries by referring to F_{k^*}. But, strictly speaking, it's not really F_{k^*} which can be consulted. We get around this as follows.

Given partial functions E and F_{k^*}, these functions having arisen in Game X, define the partial function $\widehat{F}_{k^*}$ by

$$\widehat{F}_{k^*}(x) = \begin{cases} F_{k^*}(x) & \text{if } F_{k^*}(x) \text{ is defined,} \\ E(x \oplus k_1^*) \oplus k_2^* & \text{if } E(x \oplus k_1^*) \text{ is defined, and} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Thus, in executing Game X, defining a value for E or F_{k^*} can implicitly define a new value for $\widehat{F}_{k^*}$.

At face value, the above definition might be inconsistent—this could happen if both $F_{k^*}(x)$ and $E(x \oplus k_1^*)$ are defined for some x, and with "clashing" values (ie., values which do not differ by k_2^*). Before we proceed, we observe that this can never happen:

Claim A.1 *Let E and F_{k^*} be partial functions which may arise in in Game X. Then the function $\widehat{F}_{k^*}$, as described above, is well-defined.*

The proof is by induction on the number of "Define" steps (Steps E-3, F-3, or F^{-1}-3) in the definition of Game X, where points of $\widehat{F}_{k_*}$ become defined as Game X executes. The basis (when E and F^{-1} are completely undefined) is trivial. So suppose that, in step E-3, we set $E(P) = C$. Is it possible that this definition of $E(P)$ will cause $\widehat{F}_{k_*}$ to become ill-defined? The only potential conflict is between the new $E(P)$ value and a value already selected for $F_{k^*}(P \oplus k_1^*)$. So if $F_{k^*}(P \oplus k_1^*)$ was not yet defined, there is no new conflict created in Step E-3. If, on the other hand, $F_{k^*}(P \oplus k_1^*)$ was already defined, then its value, by virtue of Step E-2, is $E(P) \oplus k_2^*$. This choice results in $\widehat{F}_{k^*}$ remaining well-defined. The analysis for the cases corresponding to Steps F-3 and F^{-1}-3 is exactly analogous, and is omitted. $\Diamond$

The function $\widehat{F}_{k^*}$, as defined for Game X, also makes sense for Game X', where $\widehat{F}_{k^*}(x) = F_{k^*}(x)$. Our strategy, then, is to explain the effect of each E, F_{k^*}, and $F_{k^*}^{-1}$ query strictly in terms of $\widehat{F}_{k^*}$. We then observe that Game X' responds to its oracle queries in an absolutely identical way. This suffices to show the games equivalent.

Case 1. We first analyze the behavior of Game X on oracle query $E(P)$. To begin, note that Game X never defines the value of $E(P)$ unless it has received P as a query. So since A never repeats queries (see the assumptions just following the theorem statement) $E(P)$ must be undefined at the time of query P. Consequently, at the time of query P, $\widehat{F}_{k^*}(P \oplus k_1^*)$ will be defined iff $F_{k^*}(P \oplus k_1^*)$ is defined, and $\widehat{F}_{k^*}(P \oplus k_1^*) = F(P \oplus k_1^*)$. *Case 1a.* When $\widehat{F}_{k^*}(P \oplus k_1^*)$ is defined, then Game X returns the value of $C = \widehat{F}_{k^*}(P \oplus k_1^*) \oplus k_2^*$. In this case, setting $E(P) = C$ leaves $\widehat{F}_{k^*}$ unchanged. *Case 1b.* When $\widehat{F}_{k^*}(P \oplus k_1^*)$ is undefined, then C is repeatedly chosen uniformly from $\overline{\text{Range}}(E)$ until $F_{k^*}^{-1}(C \oplus k_2^*)$ is undefined. By the definition of $\widehat{F}_{k^*}$ it follows that $y = C \oplus k_2^*$ is uniformly distributed over $\overline{\text{Range}}(\widehat{F}_{k^*})$. In this case, setting $E(P) = C$ sets $\widehat{F}_{k^*}(P \oplus k_1^*) = y$.

Now compare the above with Game X' on query $E(P)$. When $F_{k^*}(P \oplus k_1^*)$ is defined, then $C = F_{k^*}(P \oplus k_1^*) \oplus k_2^*$ is returned and no function values are set. When $F_{k^*}(P \oplus k_1^*)$ is undefined, y is chosen uniformly from $\overline{\text{Range}}(F_{k^*})$, $F_{k^*}(P \oplus k_1^*)$ is set to y (and implicitly $\widehat{F}_{k^*}(P \oplus k_1^*)$ is set to y), and $C = y \oplus k_2^*$ is returned. Thus, the behavior of Game X' on query $E(P)$ is identical to the behavior of Game X on query $E(P)$.

Case 2. We will be somewhat briefer with our analyses of the F and F^{-1} oracles, which are similar to the analysis above. *Case 2a.* On oracle query $F_k(x)$, when $k \neq k^*$ then the behavior of Game X is clearly identical to Game X'. *Case 2b.* When $k = k^*$ then $F_{k^*}(x)$ is defined iff a query of the form $E(x \oplus k_1^*)$ has been made. This holds iff $\widehat{F}_{k^*}(x)$ is defined (since $F_{k^*}(x)$ would not have been queried before). By a straightforward argument the value y returned from the query $F(x)$ will then be $y = E(x \oplus k_1^*) \oplus k_2^* = \widehat{F}_{k^*}(x)$ in both games. *Case 2c.* When $\widehat{F}_{k^*}(x)$ is undefined, then in both games y is uniformly chosen from $\overline{\text{Range}}(\widehat{F}_{k^*})$ and $\widehat{F}_{k^*}(x)$ is defined to be y. Thus, in all cases, Game X behaves identically to Game X'.

Case 3. Finally, on oracle query $F_k^{-1}(y)$, the case $k \neq k^*$ is again trivial. When $k = k^*$, then $\widehat{F}_{k^*}^{-1}(y)$ will be defined iff $E^{-1}(y \oplus k_2^*)$ is defined, in which case $x = E^{-1}(y \oplus k_2^*) \oplus k_1^* = \widehat{F}_{k^*}^{-1}(y)$ in both games. When $\widehat{F}_{k^*}^{-1}(y)$ is undefined, then in both games x is chosen uniformly from $\overline{\mathrm{Dom}}(\widehat{F}_{k^*})$ and $\widehat{F}_{k^*}(x)$ is defined to be y. Again, Game X behaves identically to Game X'.

Advanced Slide Attacks

Alex Biryukov* and David Wagner**

Abstract. Recently a powerful cryptanalytic tool—the slide attack—was introduced [3]. Slide attacks are very successful in breaking iterative ciphers with a high degree of self-similarity and even more surprisingly are independent of the number of rounds of a cipher. In this paper we extend the applicability of slide attacks to a larger class of ciphers. We find very efficient known- and chosen-text attacks on generic Feistel ciphers with a periodic key-schedule with four independent subkeys, and consequently we are able to break a DES variant proposed in [2] using just 128 chosen texts and negligible time for the analysis (for one out of every 2^{16} keys). We also describe *known-plaintext* attacks on DESX and Even-Mansour schemes with the same complexity as the best previously known *chosen-plaintext* attacks on these ciphers. Finally, we provide new insight into the design of GOST by successfully analyzing a 20-round variant (GOST$\oplus$) and demonstrating weak key classes for all 32 rounds.

1 Introduction

The *slide attack* is a powerful new method of cryptanalysis of block-ciphers introduced in [3]. The unique feature of this new cryptanalytic attack is its independence of the number of rounds used in the cipher of interest: when a slide attack is possible, the cipher can be broken no matter how many rounds are used. This capability is indispensable in a study of modern iterative block ciphers and hash functions. As the speed of computers grows, it is natural to use more and more rounds, which motivates our study of attacks that are independent of the number of rounds. While addition of a few rounds usually stops even a very sophisticated cryptanalytic attack (such as a differential or linear attack), in contrast a cipher vulnerable to slide attacks cannot be strengthened by increasing the number of its rounds. Instead, one must change the key-schedule or the design of the rounds.

In [3] it was shown that slide attacks exploit the degree of *self-similarity* of a block cipher and thus are applicable to iterative block-ciphers with a periodic key-schedule. It was also shown that slide attacks apply to auto-key ciphers (where the choice of the round subkeys is data-dependent). As an example an attack was presented on modified Blowfish [17], a cipher based on key-dependent S-boxes which so far had resisted all the conventional attacks.

* Applied Mathematics department, Technion - Israel Institute of Technology, Haifa, Israel 32000, and Computer Science department, The Weizmann Institute of Science, Rehovot 76100, Israel. Email: albi@wisdom.weizmann.ac.il

** University of California, Berkeley. Email: daw@cs.berkeley.edu

Table 1. Summary of our attacks on various ciphers.

Cipher	(Rounds)	Key bits	Best Previous Attack			Our Attack		
			Data	Type	Time	Data	Type	Time
2K-DES	(∞)	96	2^{32}	KP	2^{50}	2^{32}	KP	2^{33}
2K-DES	(∞)	96	2^{32}	KP	2^{50}	2^{17}	CP/CC	2^{17}
4K-Feistel	(∞)	192	—		—	2^{32}	KP	2^{33}
4K-Feistel	(∞)	192	—		—	2^{17}	CP/CC	2^{17}
4K-DES*	(∞)	192	—		—	2^{17}	CP/CC	2^{17}
Brown-Seberry-DES*	(∞)	56	—		—	128	CP/CC	2^{7}
DESX	(16)	184	2^{m}	CP	2^{121-m}	$2^{32.5}$	KP	$2^{87.5}$
DESX	(16)	184	2^{m}	CP	2^{121-m}	$2^{32.5}$	CO	2^{95}
Even-Mansour	(—)	$2n$	$2^{n/2}$	CP	$2^{n/2}$	$2^{n/2}$	KP	$2^{n/2}$
GOST$\oplus$	(20)	256	—		—	2^{33}	KP	2^{70}

CO — ciphertext-only, KP — known-plaintext, CP — chosen-plaintext, CP/CC — chosen plaintext/ciphertext. * – Our attack on 4K-DES and Brown-Seberry-DES works for $1/2^{16}$ of all keys. Note that attacks on 2K-DES work for all the keys.

The existence of attacks which are independent of the number of rounds is perhaps counter-intuitive. To illustrate this consider a quote from [15]:

> "Except in a few degenerate cases, an algorithm can be made arbitrarily secure by adding more rounds."

Slide attacks force us to revise this intuition, and this motivates our detailed study of advanced sliding techniques.

In this paper we introduce advanced sliding techniques—*sliding with a twist* and the *complementation slide*—that result in a more efficient slide attacks and allow to attack new classes of ciphers. We illustrate these techniques on generic Feistel constructions with two- or four-round self-similarity as well as a Luby-Rackoff construction and also the example ciphers 2K-DES and 4K-DES, which differ from DES only by having 64 rounds, a 96- or 192-bit key, and a simplified (periodic) key-schedule. Analysis of these ciphers is of independent interest since it demonstrates the dangers of some ways to extend DES. Specifically we show a very efficient attack on a variant of DES proposed in [2]: our attack uses only 128 chosen texts and negligible time of analysis (for a 2^{-16} fraction of all keys).

We then apply the newly developed methods to the DESX and Even-Mansour schemes, and we show known-plaintext slide attacks with the same complexity as the best previously known chosen-plaintext attacks. We also apply slide attacks to the GOST cipher (a Russian equivalent of DES) obtaining insights on its design.

See Table 1 for a summary of our results. For each cipher a number of rounds that our attack is able to cover is presented; ∞ is shown if our attack is independent of the number of rounds of a cipher. The block size in bits is denoted by n, and the 'Key bits' column denotes the number of secret key bits of the cipher.

This paper is organized as follows: In Section 2 we briefly describe conventional slide attacks. We develop several advanced sliding techniques in Section 3,

illustrating them on generic Feistel ciphers with periodic key-schedules. As a side effect we receive a distinguishing attack on the $\Psi(f, g, f, g, \ldots, f, g)$ Luby-Rackoff construction (see the end of Section 3.2). We then apply the newly developed techniques to the analysis of DESX and Even-Mansour schemes in Section 4. In Section 5 we turn advanced slide attacks to the analysis of GOST. Finally Section 6 summarizes some related work and Section 7 outlines some possible directions for further research.

2 Conventional Slide Attacks

Earlier work [3] described a simple form of slide analysis applicable to ciphers with self-similar round subkey sequences or autokey ciphers. We briefly sketch those ideas here; see [3] for full details and cryptanalysis of a number of ciphers, and Section 6 for other related work.

In the simplest case, we have an r-round cipher E whose rounds all use the same subkey, so that $E = F \circ F \circ \cdots \circ F = F^r$. Note that if the key schedule of a cipher is periodic with period p, we can consider F to be a "generalized" round consisting of p rounds of the original cipher. We call such ciphers **p-round self-similar**. Let $\langle P, C \rangle$ be a known plaintext-ciphertext pair for E. The crucial observation is

$$P' = F(P) \quad \text{implies} \quad C' = E(P') = F^r(F(P)) = F(F^r(P)) = F(C).$$

In a standard slide attack, we try to find pairs $\langle P, C \rangle$, $\langle P', C' \rangle$ with $P' = F(P)$; we call such a pair a **slid pair**, and then we will get the extra relation $C' = F(C)$ "for free."

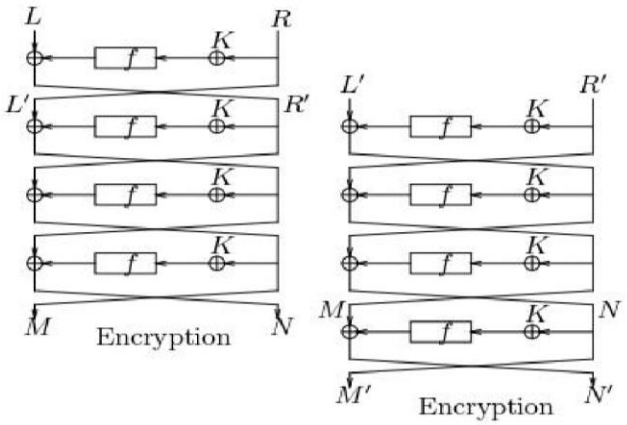


Fig. 1. A conventional slide attack on a generic Feistel cipher with one-round self-similarity. If $L' = R$ and $R' = L \oplus f(K \oplus R)$, the texts shown above will form a slid pair, and we will have $M' = N$ and $N' = M \oplus f(K \oplus N)$.

Slide attacks provide a very general attack on iterated product ciphers with repeating round subkeys. The only requirement on F is that it is very weak

against known-plaintext attack with two pairs (we are able to relax this requirement later, in Section 3.5). More precisely, we call $F_k(x)$ a **weak** permutation if given the two equations $F_k(x_1) = y_1$ and $F_k(x_2) = y_2$ it is "easy" to extract the key k. Such a cipher (with a n-bit block) can be broken with only $2^{n/2}$ known texts, since then we obtain 2^n possible pairs $\langle P, C\rangle$, $\langle P', C'\rangle$; as each pair has a 2^{-n} chance of forming a slid pair, we expect to see one slid pair which discloses the key.

Feistel ciphers form an important special case for sliding, since the attack complexity can be substantially reduced from the general case. We depict in Figure 1 a conventional slide attack on a Feistel cipher with repeating round subkeys. The Feistel round structure gives us an n-bit filtering condition on slid pairs, which lets us reduce the complexity of analysis to about $2^{n/2}$ time and space, a significant improvement over the 2^n work required for the general attack listed above. Furthermore, there is a chosen-text variation which works against Feistel ciphers with about $2^{n/4}$ chosen plaintexts: we may simply use structures to 'bypass the first round'. See [3] for details.

In this paper, we focus on generalizing the slide attack to apply to a broader range of constructions.

3 Advanced Sliding Techniques

In this section we show several ways of extending the basic slide attack to apply to larger classes of ciphers. In the following subsections we introduce two new methods: the *complementation slide* and *sliding with a twist*.

We will describe these new techniques by applying them first to a generic Feistel cipher with a 64-bit block and self-similar round subkeys. (See Figure 1 for an example of such a cipher, where the subkeys exhibit one-round self-similarity. In this section, we consider up to four-round self-similarity.) For ease of illustration we will show graphically ciphers with only a small number of rounds, but we emphasize that the attacks described in this section apply to ciphers with any number of rounds. After describing the basic attack techniques we will show how to extend them to real ciphers.

3.1 The Complementation Slide

First we show a method to amplify self-similarity of Feistel ciphers with two-round self-similarity by exploiting its complementation properties, thus allowing for much better attacks. We call this approach the *complementation slide*.

In the conventional attack, to deal with two-round self-similarity one must slide by two rounds (thus achieving a perfect alignment of rounds with K_0 and K_1), but this yields inefficient attacks. In contrast, we suggest to slide by only one round. This introduces the difference $\Delta = K_0 \oplus K_1$ between slid encryptions in all the rounds. Notice that we have effectively amplified the self-similarity of the cipher from 2-round to 1-round self similarity. However together with amplified

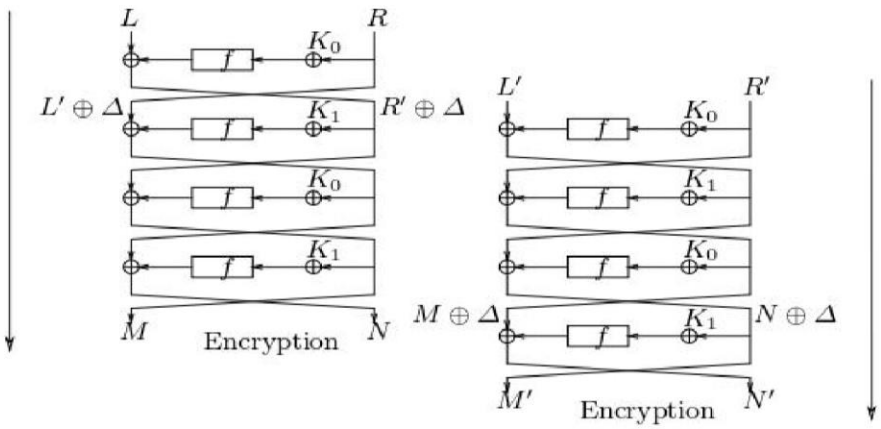


Fig. 2. A *complementation slide* attack on a Feistel cipher with two-round self-similarity. If $L' = R \oplus \Delta$ and $R' = L \oplus f(K_0 \oplus R) \oplus \Delta$, the texts shown above will form a slid pair, and we will have $M' = N \oplus \Delta$ and $N' = M \oplus f(K_1 \oplus N \oplus \Delta) \oplus \Delta$, where $\Delta = K_0 \oplus K_1$.

self-similarity we have introduced differences between rounds of encryption in a slid pair. How can the attack proceed?

Our answer is to choose a slid pair so that the plaintext differences will cancel the difference between the subkeys. Instead of searching for plaintexts with slid difference zero, we search for plaintexts with slid difference $\langle \Delta, \Delta \rangle$. (Note: We say that a pair of plaintexts P, P' has *slid difference* d if $F(P) \oplus P' = d$.) Such a slid difference will propagate with probability one through all the rounds, and thus will appear at the ciphertext. See Figure 2 for a pictorial illustration of the attack.

The slid pairs can be found in a pool of 2^{32} known plaintexts, as before. If we denote the plaintext by $P = \langle L, R \rangle$ and the ciphertext by $C = \langle M, N \rangle$, we get the following slid equations:

$$\langle L', R' \rangle = \langle R, L \oplus f(K_0 \oplus R) \rangle \oplus \langle \Delta, \Delta \rangle$$
$$\langle M', N' \rangle = \langle N, M \oplus f(K_1 \oplus N \oplus \Delta) \rangle \oplus \langle \Delta, \Delta \rangle.$$

Thus we have $L' \oplus M' = R \oplus N$ which is a 32-bit condition on a slid pair. Moreover the second equation suggests a 32-bit candidate for $\Delta = K_0 \oplus K_1$; if we have several slid pairs, this value should coincide for all of them (although we do not need the latter property in our attack). Thus the S/N ratio of this attack is very high. As soon as one slid pair is found, we derive $\Delta = K_0 \oplus K_1$. Then, if the round function f is weak enough, we will be able to derive the keys K_0 and K_1 themselves from the first and second equations. We will only need to examine 2^{31} pairs (due to the 32-bit filtering condition) and each pair suggests at most one candidate key, so the work-factor of the attack is very low.

To summarize, this gives a known plaintext attack on a generic Feistel cipher with two-round self-similarity. The complexity of the attack is quite realistic: we need just 2^{32} known texts and at most 2^{32} light steps of analysis. However, see Section 3.2 for an even better attack.

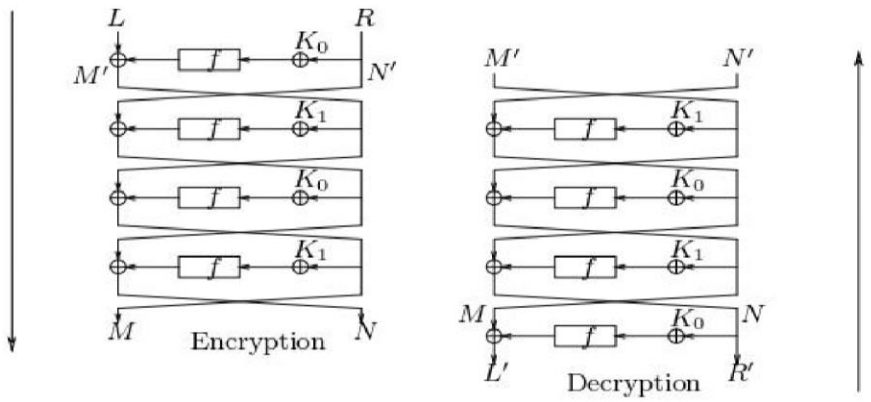


Fig. 3. *Sliding with a twist*, applied to a Feistel cipher with two-round self-similarity. If $N' = R$ and $M' = L \oplus f(K_0 \oplus R)$, the texts shown above will form a (twisted) slid pair, and we will have $R' = N$ and $L' = M \oplus f(K_0 \oplus N)$.

Even more interestingly: We can consider a variant with four independent subkeys, K_0, K_1, K_2, K_3, so that the key size is 128 bits. If we slide by two rounds we find that the XOR differences between subkeys are 2-round self-similar! A modified version of the above attack works, although the S/N ratio is not as high as before. Complementation sliding thus provides a powerful technique for amplifying self-similarity in iterated ciphers.

3.2 Sliding with a Twist

We next describe a novel technique of sliding with a *twist* on a Feistel cipher with two-round self-similarity. This allows for even better attacks than those presented above. See also our attack on DESX in Section 4 for an important application of sliding with a twist.

If we ignore the final swap for the moment, then decryption with a Feistel cipher under key K_0, K_1 is the same as encryption with key K_1, K_0[1]. Of course, Feistel encryption with key K_0, K_1 is very similar to encryption with key K_1, K_0: they are just out of phase by one round. Therefore, we can slide by one round a decryption process against an encryption process (the *twist*). This provides us with a slid pair with an overlap of all rounds except for one round at the top and one round at the bottom. Notice that due to the twist these rounds both use the same subkey K_0. See Figure 3 for a graphical depiction.

The attack begins by obtaining a pool of 2^{32} known texts, so that we expect to find one slid pair. For a slid pair, we have

$$\langle M', N' \rangle = \langle L \oplus f(K_0 \oplus R), R \rangle \qquad \langle L', R' \rangle = \langle M \oplus f(K_0 \oplus N), N \rangle$$

which gives us a 64-bit filtering condition on slid pairs (namely $N' = R$ and $R' = N$). Thus the slid pair can be easily found with a hash table and 2^{32} work, and it immediately reveals the subkey K_0.

[1] In [3] such cipher, based on DES was called 2K-DES.

The rest of the key material can be obtained in a second analysis phase with a simplified conventional sliding (by two rounds and without a twist) using the same pool of texts and with less than 2^{32} work. Pick a ciphertext from a pool, partially encrypt it with K_0 and search the pool of ciphertexts for one with coinciding 32 bits. If such a ciphertext is found perform a similar check on their plaintexts. If both conditions hold this is a slid pair that provides us with K_1. This attack requires just 2^{32} known texts and 2^{33} work.

Moreover, there is a chosen-plaintext/ciphertext variant that allows us to reduce the number of texts down to 2^{17} with the use of structures. We generate a pool of 2^{16} plaintexts of the form (L_i, R) and obtain their encryptions. Also, we build a pool of 2^{16} ciphertexts of the form (M'_j, N') and decrypt each of them, where the value $N' = R$ is fixed throughout the attack. This is expected to give one slid pair, and then the analysis proceeds as before.

This demonstrates that sliding with a twist is capable of attacking any n-bit Feistel block cipher with a two-round periodic key-schedule with $2^{n/2}$ known plaintexts and about $2^{n/2}$ time, or with about $2^{n/4}$ chosen plain-ciphertexts and about $2^{n/4}$ time. Also, sliding with a twist can be used to distinguish a Luby-Rackoff [13] construction with two alternating pseudo-random functions f and g and with an arbitrary number of rounds (an accepted notation is $\Psi(f, g, f, g, \ldots, f, g)$) from a random permutation with about $2^{n/2}$ known plaintexts and similar time (given that the block size is n bits), or with about $2^{n/4}$ chosen plaintext/ciphertext queries and similar time.

3.3 Better Amplification of Self-Similarity: Four-Round Periodicity

In this section we combine the *complementation slide* and *sliding with a twist* to amplify the self-similarity of round subkeys even further. Consider a Feistel cipher with key schedule that repeats every four rounds, using independent subkeys K_0, K_1, K_2, K_3, and suppose these keys are XORed at the input of the f-function. We call this generic cipher a *4K-Feistel* cipher.

One may naively slide by two rounds to amplify self-similarity, like this:

$$\begin{array}{l} K_0\ K_1\ K_2\ K_3\ K_0\ K_1\ \ldots \\ \qquad K_0\ K_1\ K_2\ K_3\ K_0\ K_1\ \ldots \end{array}$$

Then one may use a complementation slide technique using the slid difference $\langle K_1 \oplus K_3, K_0 \oplus K_2 \rangle$. However, there doesn't seem to be any way to make this attack work with less than $2^{n/2}$ texts, and the analysis phase is hard.

Better results are possible if one applies *sliding with a twist*. At a first glance, the twist may not seem to be applicable, but consider combining it simultaneously with the *complementation slide*, like this:

$$\begin{array}{l} K_0\ K_1\ K_2\ K_3\ K_0\ K_1\ K_2\ K_3\ K_0\ \ldots \\ \quad K_3\ K_2\ K_1\ K_0\ K_3\ K_2\ K_1\ K_0\ K_3\ \ldots \end{array}$$

The top row represents an encryption, and the bottom represents a decryption (or, equivalently, encryption by K_3, K_2, K_1, K_0, due to the similarity between encryption and decryption in Feistel ciphers).

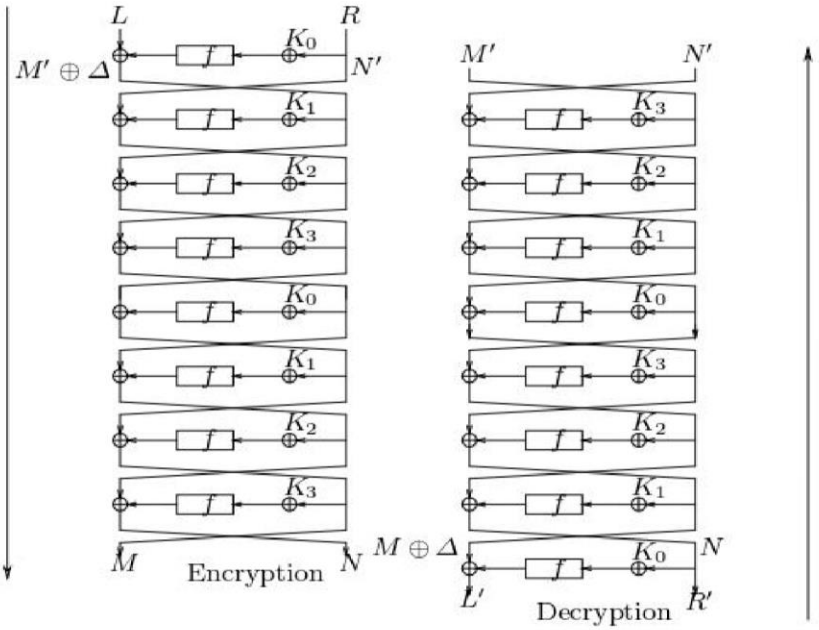


Fig. 4. Combining the complementation slide and sliding with a twist techniques in a single unified attack against a Feistel cipher with four-round self-similarity.

Now note that the odd rounds always line up, but the even rounds have the constant difference $K_1 \oplus K_3$ in the round subkeys. Therefore, we can apply the *complementation slide* technique, if we can get texts with a slid difference of $\langle 0, K_1 \oplus K_3 \rangle$. Then we get the attack shown in Figure 4.

Combining the two advanced sliding techniques provides a number of significant benefits. First, we obtain an n-bit filtering condition, so detecting slid pairs becomes easy. Consequently, the analysis phase is straightforward. Also, the combined approach makes it easier to recover key material from a slid pair. Finally, perhaps the most important improvement is that now we can reduce the data complexity of the attack to just $2^{n/4}$ texts, in the case where chosen-plaintext/ciphertext queries are allowed. Neither advanced sliding technique can—on its own—provide these advantages; in this respect, the whole is greater than the sum of the parts.

3.4 Attack on DES with Brown-Seberry Key-schedule

In [2] an alternative key-schedule for DES was proposed. This key-schedule was supposed to be "as effective as that used in the current DES" and was "suggested for use in any new algorithm" [2]. This variant of DES was already studied in [1] resulting in a related-key attack on it. In this section we show a chosen plaintext/ciphertext slide attack on this variant of DES, which uses only 128 chosen texts and negligible time for analysis. The attack works for 2^{40} out of 2^{56} keys.

To remind the reader: the DES key-schedule consists of two *permuted-choice* permutations PC1 and PC2, and a rotation schedule. The first permuted choice PC1 is used to reduce the key-size from 64 bits to 56 bits. Then the result is divided into two 28-bit registers C and D. Each round we cyclicly rotate both registers by one or two bits to the left. Permuted choice PC2 is applied to the result, which picks 24 bits from each 28-bit register and thus forms a 48-bit round subkey.

In [2] a key-schedule that rotates by 7 bits every round was proposed (instead of the irregular 1,2-bit rotations used in DES). Due to a larger rotation amount which spreads bits between different S-boxes the PC2 permutation was simplified to become an identity permutation which just discards the last 4 bits of each 28-bit register. We claim that for $1/2^{16}$ of the keys, this variant can be broken with our sliding with a twist techniques as follows: the known-plaintext attack will require $2^{32.5}$ texts, time and space; the chosen-plaintext/ciphertext, however, will require only 2^7 texts!

First of all notice that since the new rotation amount (7 bits) divides the size of the key-schedule registers (28 bits) the registers C, D return to their original state every four rounds. This results in a key-schedule with a period of four, which can be analyzed by the methods that we developed in the previous sections for the four-round self-similar Feistel ciphers. We will extend the standard attack even further by noticing that DES key-schedule is used and not four independent round subkeys as in our previous model. However, DES-like ciphers introduce one small complication: the DES round function XORs the subkey against the 48-bit expanded input rather than the raw 32-bit input, so the complementation slide only works if the 48-bit subkey difference is expressible as the expansion of some 32-bit text difference.

Let $J_i = \langle C \lll 7i, D \lll 7i \rangle$ so that $K_i = \mathrm{PC2}(J_i)$. For the sliding with a twist to work in the case of DES we need $K_1 \oplus K_3$ to have an 'expandable' form in order to pass through the 32 to 48 expansion of the DES round function. Note also that if $J_1 = \langle u, v, u', v' \rangle$ where u, v, u, v' are all 14-bit quantities, then $J_3 = \langle v, u, v', u' \rangle$ in a Brown-Seberry key-schedule, and thus for $Z = J_1 \oplus J_3$ we have $Z_i = Z_{i+14}$ for $i \in \{0, 1, \ldots, 13, 28, 29, \ldots, 41\}$. The PC2 just discards Z_i for $i \in \{24, 25, \ldots, 27, 52, 53, \ldots, 55\}$ to get the 48-bit quantity $Y = \mathrm{PC2}(Z) = K_1 \oplus K_3$.

If we insist $Y = \mathrm{Expansion}(X)$ for some X, we get 16 constraints on Y: namely, $Y_i = Y_{i+2}$ for $i = 6j + k$, $j \in \{0, \ldots, 7\}$, $k \in \{4, 5\}$ where subscripts are taken modulo 48. Thus we have

$$Z_i = Z_{i+2} \text{ for } i \in \{4, 5, 10, 11, 16, 17, 32, 33, 38, 39, 44, 45\};$$

and $Z_i = Z_{i+6}$ for $i \in \{22, 23, 50, 51\}$. Therefore $Y = K_1 \oplus K_3$ is expandable if and only if $Z = J_1 \oplus J_3$ has the form

$$Z = \langle abcdcd\ cdefgh\ ghabcd\ cdcdef\ ghgh\ \ efklkl\ klabmn\ mnefkl\ klklab\ mnmn \rangle$$

where $a, b, .., n$ are 12 arbitrary bits. we see that there are exactly 2^{12} expandable values of $K_1 \oplus K_3$ that satisfy the required constraints. Moreover, for each expandable value of $K_1 \oplus K_3$, there are 2^{28} possible values of J_1 for which $K_1 \oplus K_3$

has the given value (since we may choose u and u' arbitrarily, setting v and v' as required to ensure that $\langle u \oplus v, u \oplus v, u' \oplus v', u' \oplus v' \rangle$ has an appropriate value for $J_1 \oplus J_3$).

This shows that there are 2^{40} values of J_1 that lead to four-round self-similarity with an expandable value for $K_1 \oplus K_3$. In other words, $1/2^{16}$ of the keys are breakable with our standard attack. Note that the standard attack for the case of four independent round subkeys uses $2^{32.5}$ known texts, time and space, or 2^{17} chosen texts, time and space. However, we may use the special structure of $K_1 \oplus K_3$ to significantly reduce the complexity of the chosen-text attack.

In particular, we choose 2^6 plaintexts of the form $\langle L_i, R \rangle$ and 2^6 ciphertexts of the form $\langle M'_j, N' \rangle$, where $R = N'$ is fixed throughout the attack and

$$\begin{aligned} L_i &= \langle bcdc\ def0\ 0abc\ dcde\ f000\ 0ab0\ 0ef0\ 000a \rangle \\ M'_j &= \langle 0000\ 000g\ h000\ 0000\ 0klk\ l00m\ n00k\ lkl0 \rangle, \text{ so that} \\ L_i \oplus M'_j &= \langle bcdc\ defg\ habc\ dcde\ fklk\ labm\ nefk\ lkla \rangle \end{aligned}$$

and thus $\text{Expansion}(L_i \oplus M'_j) = K_1 \oplus K_3$ for some i, j, which immediately gives us a slid pair. (We assume for ease of description that the cipher includes the final swap and no IP or FP, so that Figure 4 in Section 3.2 applies.) We can recognize the slid pair by a 64-bit filtering condition on $\langle M, N \rangle, \langle L', R' \rangle$, and so the analysis phase is easy.

To sum up, this provides an attack on the cipher that breaks $1/2^{16}$ of the keys with 2^7 chosen texts, time and space.

3.5 Generalizations for a Composition of Stronger Functions

In Section 2 we have seen how a typical slide attack may work. However, in many cases this approach is too restrictive, since it may be desirable to analyze ciphers which decompose into a product of stronger functions; in particular, the round function may be strong enough that multiple input/output pairs are required to recover any key material. In this section we show several techniques to handle this situation.

One approach is to use a differential analysis. Denote by n the block size of the cipher. Suppose there is a non-trivial differential characteristic $\Delta X \rightarrow \Delta Y$ of probability p for the round function. We associate to each plaintext P the plaintext $P \oplus \Delta X$ and to each plaintext P' another plaintext $P' \oplus \Delta Y$. Then, if $P' = F(P)$, we will also have $P' \oplus \Delta Y = F(P \oplus \Delta X)$ with probability p (thanks to the characteristic $\Delta X \rightarrow \Delta Y$), which provides two slid pairs. In this way we may obtain four known input/output pairs for the function F. We can generate a set of $3 \cdot 2^{n/2} p^{-1/2}$ chosen plaintexts such that for plaintext P in the chosen set the plaintexts $P \oplus \Delta X$ and $P \oplus \Delta Y$ are also in the set; then we will expect to see one pair P, P' satisfying both the slide and the differential patterns.

The second approach (which is probably the simplest) works like this. Suppose to recover the key we need N known texts for the round function F. For

each plaintext P, we suggest to get the encryption $E(P)$ of P, and the double-encryption $E^2(P) = E(E(P))$ of P, and so on, until we have obtained $E^{2N}(P)$. Then, if $P' = F(E^i(P))$, we find $2N - i$ slid pairs "for free" by the relation $E^j(P') = F(E^{j+i}(P))$ for $j = 1, .., 2N - i$. With $2^{(n+1)/2}N^{1/2}$ chosen texts, we expect to find about N slid pairs in this way (probably all in the same batch formed from a single coincidence of the form $P' = F(E^i(P))$). To locate the batch of slid pairs, one could naively try all 2^{n+2} possible pairings of texts (though in practice we would search for a more efficient approach); each pairing that gives N or more known texts for F will suggest a key value that can then be tested[2].

Normally this last attack would be classified as an adaptive chosen-plaintext attack. However, note that in many modes (CBC, CFB) it can be done with a non-adaptive chosen-plaintext attack. Furthermore, in the case of OFB mode, a known plaintext assumption suffices. However, these comments assume that re-encryption preserves the sliding property, which is not always the case.

Another possible generalization is in the case of Feistel-ciphers. In this case one can detect slid pairs even before trying to find the correct secret key k. In the case of a balanced Feistel cipher with block size n we have an $n/2$-bit condition on the ciphertexts of a slid pair. This increases the S/N ratio considerably, filtering out most of the incorrect pairs even before we start the analysis. This property allows an attacker to accumulate sufficient number of slid pairs before he starts an attack on a round-reduced variant of a cipher.

Notice also that if we use a technique for receiving many slid pairs in the case of a Feistel-cipher, we would need only $2 \cdot 2^{n/4}N$ chosen texts, and the S/N ratio will be excellent by comparing several halves of the ciphertexts.

Furthermore if $N^{1/2} > 2^{n/4}$, an absolutely different idea can be used. Choose a random starting point P. About $2^{n/2}$ times iterate the following operation $s \circ E$, where s denotes swap of the halves (the swap is needed only if E has no final swap at the last round). This way one can obtain more than $2^{n/2-\log r}$ slid pairs (here r denotes the number of rounds of a cipher). The S/N ratio is again excellent. The idea is that we essentially search for a symmetric point (A, A) of a round function, which happens after about $2^{n/2}$ rounds ($2^{n/2-\log r}$ encryptions). This does not necessarily happen in the middle of a cipher, so we may have to perform up to r times more encryptions before we reach a fixed point for E. In half of the cases (if the first symmetric point happened at an even round) we will receive an orbit "slidable" by two rounds, and in other half of the cases (symmetric point at odd rounds) an orbit will be "slidable" by one round. Even if an orbit is "slidable" only by two, and thus $n/2$-bit filtration will be unreachable to us, the encryption fixed point that ends our orbit helps us slide the orbit correctly (at most $r/2$ possibilities).

[2] If E were behaving like a random function, it would be enough to take $2^{n/2} + N$ encryptions, from an orbit of some arbitrarily chosen element P, but since E is expected to behave like a random permutation, an orbit of P will be a part of usually a very large cycle, leaving no place for collisions. Considering a few more orbits will not help either.

4 Cryptanalysis of DESX and Even-Mansour Schemes

DESX is an extension of DES proposed by Rivest in 1984. It makes DES more resistant to exhaustive search attacks by XORing two 64-bit keys: one at the input and another at the output of the DES encryption box[3]. See [10, 16] for theoretical analysis of DESX.

In this section we show the unexpected result that the DESX construction contains just enough symmetry to allow for slide attacks. These results are actually generally applicable to all uses of pre- and post-whitening (when applied using self-inverse operations like XOR), but for convenience of exposition we will focus on DESX.

The attacks presented here are another example of an application of the powerful new *sliding with a twist* technique. Our attacks on DESX are significantly better than the best previously known attacks: we need just $2^{32.5}$ *known* texts and $2^{87.5}$ time for the analysis, while the best generic attack reported in the literature is a *chosen*-plaintext attack with comparable complexity [10, 16][4]. Thus, sliding techniques allow one to move from the chosen-text attack model to the more realistic known-text attack model. Even more unexpectedly, our attack can also be converted to a ciphertext-only attack.

We briefly recall the definition of DESX. Let $E_k(x)$ denote the result of DES-encrypting the plaintext x under the key k. Then we define DESX encryption under the key $K = \langle k, k_x, k_y \rangle$ as $EX_K(p) = k_y \oplus E_k(p \oplus k_x)$. To set up the necessary slide relation, we imagine lining up a DESX encryption against a slid DESX decryption, as shown in Figure 5. More specifically, we say that the two known plaintext pairs $\langle p, c \rangle$ and $\langle p', c' \rangle$ form a slid pair if $c \oplus c' = k_y$. Consequently, for any slid pair, we will have

$$p' = k_x \oplus E_k^{-1}(c' \oplus k_y) = k_x \oplus E_k^{-1}(c)$$

as well as $p = k_x \oplus E_k^{-1}(c')$. Combining these two equations yields $k_x = p \oplus E_k^{-1}(c') = p' \oplus E_k^{-1}(c)$. As a result, we get a necessary property of slid pairs: they must satisfy

$$E_k^{-1}(c) \oplus p = E_k^{-1}(c') \oplus p'. \qquad (*)$$

To get a single slid pair, we obtain $2^{32.5}$ known plaintexts $\langle p_i, c_i \rangle$ and search for a pair which satisfies the sliding condition $(*)$. The pairs can be recognized efficiently with the following technique. We guess the DES key k. Next, we insert $E_k^{-1}(c_i) \oplus p_i$ into a lookup table for each i; alternatively, we may sort the texts by this value. A good slid pair $\langle p, c \rangle, \langle p', c' \rangle$ will show up as a collision in the table. Also, each candidate slid pair will suggest a value for k_x and k_y as above

[3] Note that an idea to use simple keyed transformations around a complex mixing transform goes back to Shannon [18, pp.713].

[4] One may apply differential or linear cryptanalysis to DESX, but then at least 2^{60}–2^{61} texts are needed [11]. In contrast, slide attacks allow for a *generic* attack with a much smaller data complexity.

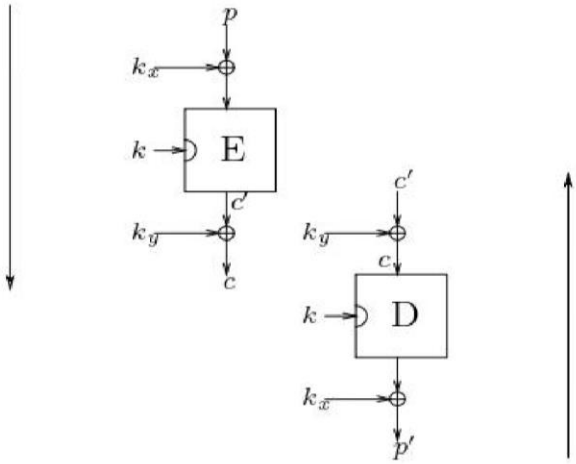


Fig. 5. Sliding with a twist, applied to DESX.

(e.g., $k_y = c \oplus c'$ and $k_x = p \oplus E_k^{-1}(c')$), so we try the suggested DESX key $\langle k, k_x, k_y \rangle$ immediately on a few known texts. With $2^{32.5}$ known texts, we expect to find one false match (which can be eliminated quickly) per guess at k, as well as one correct match (if our guess at k was correct). If this attack sketch is not clear, see the algorithmic description in Figure 6.

In total, the average complexity of our slide attack on DESX is $2^{87.5}$ offline trial DES encryptions, $2^{32.5}$ known texts, and $2^{32.5}$ space. The slide attack is easily parallelized. Compare this to the best attack previously reported in the open literature, which is a *chosen*-plaintext attack that needs 2^{121-m} time (average-case) when 2^m texts are available [10, 16]. Therefore, our attack converts the chosen-plaintext assumption to a much more reasonable known-plaintext assumption at no increase in the attack complexity.

CIPHERTEXT-ONLY ATTACKS. Note that in many cases our slide attack on DESX can even be extended to a ciphertext-only attack. We suppose (for simplicity) that most plaintext blocks are composed of just the lowercase letters 'a' to 'z', encoded in ASCII, so that 24 bits of each plaintext are known[5]. For each i we calculate 24 bits of $E_k^{-1}(c_i) \oplus p_i$ and store the result in a lookup table. Due to the weak filtering condition, by the birthday paradox we expect to find about $2^{2 \cdot 32.5-1}/2^{24} = 2^{40}$ collisions in the table. Each collision suggests a value for k_y (as $k_y = c \oplus c'$) and for 24 bits of k_x, which we immediately try with a few DESX trial decryptions on other known ciphertexts. Therefore, for each guess of k the workfactor is 2^{40} DES operations.

This provides a simple ciphertext-only attack needing about $2^{32.5}$ ciphertexts and 2^{95} offline DES operations. The work-factor can be reduced somewhat to 2^{95} simple steps (where each step is much faster than a trial decryption), if 2^{33}

[5] The attack degrades gracefully if our model of the plaintext source is only probabilistic: for instance, if half of the texts follow the model, the attack will need only $\sqrt{2}$ times as many ciphertexts and only twice as much work.

ATTACK:
1. Collect $2^{32.5}$ known plaintexts $\langle p_i, c_i \rangle$.
2. For each $k \in \{0,1\}^{56}$, do
3. Insert $\langle E_k^{-1}(c_i) \oplus p_i, i \rangle$ into a hash table keyed by the first component.
4. For each $i \neq j$ with $E_k^{-1}(c_i) \oplus p_i = E_k^{-1}(c_j) \oplus p_j$, do
5. Set $k_y = c_i \oplus c'_j$ and $k_x = p_i \oplus E_k^{-1}(c_i \oplus k_y)$.
6. Test the validity of the guessed key $\langle k, k_x, k_y \rangle$ on a few more known texts.

Fig. 6. The DESX slide attack, in full detail. It is clear that—once discovered—the attack may be described without reference to sliding, but the *sliding with a twist* methodology made it possible to find the attack in the first place.

known ciphertexts are available, by considering candidate slid pairs two at a time and filtering on the suggested value of k_y, since then the correct value of k_y will be suggested at least twice and can therefore be recognized in this way before doing any trial decryptions. Note that these ciphertext-only attacks are applicable not only to ECB mode but also to most of the standard chaining modes, including CBC and CFB modes.

CRYPTANALYSIS OF THE EVEN-MANSOUR SCHEME. In [7], Even and Mansour studied a simple n-bit block cipher construction based on a fixed pseudo-random permutation and keyed n-bit XORs at the input and at the output. Due to the generic nature of our previous attack on DESX it can also be used to analyze the Even-Mansour construction[6]. In the case of Even-Mansour we replace E_k with an unkeyed mixing transformation E on n-bit blocks, so our slide attack succeeds with just $2^{(n+1)/2}$ known plaintexts and $2^{(n+1)/2}$ work. This provides a known-plaintext attack with the same complexities as the best previously-known chosen plaintext attack [6] and within a factor of $\sqrt{2}$ away from the Even-Mansour lower bound.

5 Analysis of GOST

GOST, the Russian encryption standard [19], was published in 1989.[7] Even after considerable amount of time and effort, no progress in cryptanalysis of the standard was made in the open literature except for a brief overview of a GOST structure in [4] and a related key attack in [9]. In this section we apply slide techniques to GOST and thus are able to produce cryptanalytic results that shed some light on its internal structure.

The GOST encryption algorithm is a block cipher with 256-bit keys and a 64-bit block length. GOST is designed as a 32-round Feistel network, with 32-bit round subkeys. See Figure 7 for a picture of one round of GOST.

[6] Of course, these attacks will apply with the same complexity to DESX when the DES key k is known somehow.

[7] It was translated into English in 1993 and since then became well known to open cryptographic community.

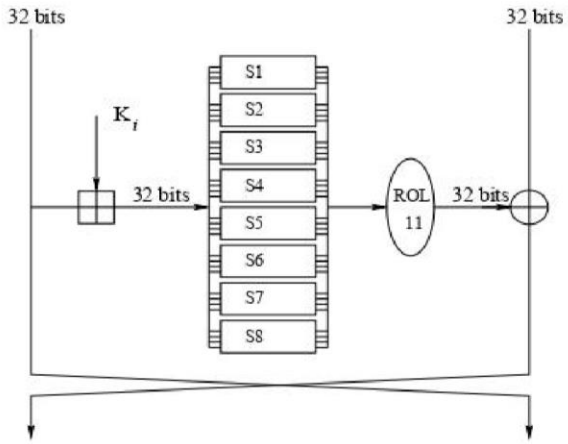


Fig. 7. One round of a GOST cipher.

The key schedule divides the 256-bit key into eight 32-bit words $K_0, \ldots, K_7$, and then uses those key words in the order $K_0, \ldots, K_7$, $K_0, \ldots, K_7$, $K_0, \ldots, K_7$, $K_7, K_6, \ldots, K_0$. Notice the 'twist' in the last 8 rounds.

THE ANALYSIS OF GOST. GOST looks like a cipher that can be made both arbitrarily strong or arbitrarily weak depending on the designer's intent since some crucial parts of the algorithm are left unspecified. A huge number of rounds (32) and a well studied Feistel construction combined with Shannon's substitution-permutation sequence provide a solid basis for GOST's security. However, as in DES everything depends on the exact choice of the S-boxes and the key-schedule. This is where GOST conceptually differs from DES: the S-boxes are not specified in the standard and are left as a secondary key common to a "network of computers"[8].

The second mystery of GOST is its key-schedule. It is very simple and periodic with the period of eight rounds except for the last eight rounds where a twist happens. It is intriguing to find a reason for the twist in the last eight rounds of the key schedule. Moreover, in many applications we may wish to use shorter 64- or 128-bit keys, yet it is not clear how to extend these to a full 256-bit GOST key securely (fill the rest with zeros, copy the bits till they cover 256 bits, copy bits in a reversed order?).

WHY THE TWIST? Consider a GOST cipher with a homogeneous key schedule, i.e., omitting the final twist (let us denote it GOST-H). Is this cipher less secure than GOST? We argue that, if one takes into account the slide attacks, it is. GOST-H can be decomposed into four identical transforms, each consisting of eight rounds of GOST. Furthermore, if one assumes that the round subkey is XORed instead of being ADDed, the cipher will have 2^{128} weak keys of the form $\langle A, B, C, D, A, B, C, D \rangle$ (here each letter represents a 32-bit GOST subkey). These keys are weak since they allow for a *sliding with a twist* attack.

[8] Contrary to common belief, the standard does not even require the S-boxes to be permutations.

There is a known plaintext attack with 2^{32} texts and time, and a chosen plaintext attack with 2^{16} texts and time; see Section 3.3 for more details.

Notice that the 2^{128} keys of the form $\langle A, B, C, D, D, C, B, A\rangle$ are also weak since GOST-H with these keys is an involution and thus double encryption will reveal the plaintext. Since these keys are invariant under a twist the same property holds for GOST itself. Also, there are 2^{32} fixed points for each key of this form, which demonstrates that there may be problems with using GOST to build a secure hash function.

THE ATTACK ON 20 ROUNDS OF GOST⊕. Suppose again that the round subkey is XORed instead of being ADDed, (we will denote this variant of GOST as GOST⊕). Here we show an application of *sliding with a twist* which results in an attack on the last 20 rounds of GOST⊕.

Applying *sliding with a twist*, we get a picture that looks like this:

$$K_4\ K_5\ K_6\ K_7\ K_0\ K_1\ K_2\ K_3\ K_4\ K_5\ K_6\ K_7\ K_7\ K_6\ K_5\ K_4\ K_3\ K_2\ K_1\ K_0$$
$$K_0\ K_1\ K_2\ K_3\ K_4\ K_5\ K_6\ K_7\ K_7\ K_6\ K_5\ K_4\ K_3\ K_2\ K_1\ K_0\ K_7\ K_6\ K_5\ K_4.$$

Let F denote 4 rounds of GOST⊕ with key $K_4, \ldots, K_7$. With a pool of 2^{33} known texts, we expect to find two slid pairs, and each slid pair gives two input/output pairs for F. Breaking F with two known texts is straightforward, and can be performed in time comparable to about 2^9 evaluations of 4-round GOST (equivalent to 2^5 20-round trial encryptions). Thus in our attack we examine all 2^{65} text pairs; each pair suggests a value for 128 bits of key material, which we store in a hash table (or sorted list). The right key will be suggested twice, so we expect to be able to recognize it easily. By the birthday paradox, there will be only about two false matches, and they can be eliminated in the next phase.

Once we have recovered $K_4, \ldots, K_7$, it is easy to learn the rest of the key in a second analysis phase. For example, we can peel off the first four rounds and look for fixed points in the same pool of texts. Since the round subkeys are palindromic in the last sixteen rounds of GOST, there are 2^{32} fixed points, and each has the value $\langle x, x\rangle$ before the last eight rounds of encryption. Thus, given a fixed point, we can try the 2^{32} values of $\langle x, x\rangle$, encrypt forward and backward eight rounds, and obtain two candidate input/output pairs for 4 rounds of GOST⊕ with key $K_0, \ldots, K_3$, so that a value for $K_0, \ldots, K_3$ is suggested after 2^5 work; then the suggested 256-bit key value is tried on another known text pair.

In all, this gives an attack on the last 20 rounds of GOST⊕ that needs 2^{33} known texts, 2^{70} work, and 2^{65} space to recover the entire 256-bit key. Note that this attack is generic and works for any set of (known) S-boxes. The large memory requirements make the attack highly impractical, but we view it as a first step towards a better understanding of the GOST design.

6 Related Work

The first step in the "sliding" direction can be dated back to a 1978 paper by Grossman and Tuckerman [8], which has shown how to break a weakened Feistel

cipher[9] by a chosen plaintext attack, independent of the number of rounds. We were also inspired by Biham's work on related-key cryptanalysis [1], and Knudsen's early work [12].

Some related concepts can be found in Coppersmith's analysis of fixed points in DES weak keys and cycle structure of DES using these keys [5]. This analysis was continued further by Moore and Simmons [14]. For a DES weak key, all round subkeys are constant, and so encryption is self-inverse and fixed points are relatively common: there are precisely 2^{32} fixed points. Note that this property will also be found in any Feistel cipher with *palindromic* round key sequences, so the slide attack is not the only weakness of ciphers with self-similar round subkey sequences.

7 Discussion

In this section we discuss possible extensions of slide attacks presented in this paper and possible directions of future research.

The most obvious type of slide attack is usually easy to prevent by destroying self-similarity in iterative ciphers, for example by adding iteration counters or fixed random constants. However more sophisticated variants of this technique are harder to analyze and to defend against. This paper is a first step towards advanced slide attacks which can penetrate more complex cipher designs.

One promising new direction is the *differential slide attack*. By sliding two encryptions against each other, we obtain new differential relations which in some cases are not available in the conventional differential analysis of a cipher. These might be very powerful, since they might for example violate the subtle design constraints placed on the system by its designer and thus result in unexpected differential properties. If key-scheduling is not self-similar or symmetric, differences in subkeys can cause constant XOR values to be introduced in the middle of the encryption process when slid pairs are considered. (In many cases, one can slide by different numbers of rounds and thus control the differences to some extent.) The drawback of this method is the same as in conventional methods: its complexity increases fast with the number of rounds, contrary to the general sliding technique, which works for arbitrary number of rounds.

Acknowledgments

We would like to thank Eli Biham for pointing to us the Brown-Seberry variant of DES key-schedule.

References

1. E. Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, J. of Cryptology, Vol.7, pp.229–246, 1994.

[9] They analyzed an 8-round Feistel cipher with eight bits of key material per round used to swap between two S-boxes S_0 and S_1 in a Lucifer-like manner: a really weak cipher by modern criteria.

2. L. Brown, J. Seberry, *Key Scheduling in DES Type Cryptosystems*, proceedings of AUSCRYPT'90, LNCS 453, pp.221–228, Springer Verlag, 1990
3. A. Biryukov, D. Wagner, *Slide Attacks*, proceedings of FSE'99, LNCS 1636, pp.245–259, Springer Verlag, 1999.
4. C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini, Y. Zheng, *Comments on Soviet Encryption Algorithm*, proceedings of EUROCRYPT'94, LNCS 950, pp.433–438, Springer Verlag, 1994.
5. D. Coppersmith, *The Real Reason for Rivest's Phenomenon*, proceedings of CRYPTO'85, pp.535–536, Springer Verlag, 1986.
6. J. Daemen, *Limitations of the Even-Mansour Construction*, proceedings of ASIACRYPT'91, pp.495–498, Springer-Verlag 1992.
7. S. Even, Y. Mansour, *A Construction of a Cipher from a Single Pseudorandom Permutation*, Journal of Cryptology, Vol.10, No.3, pp.151–161, 1997.
8. E. K. Grossman, B. Tucherman, *Analysis of a Weakened Feistel-like Cipher*, 1978 International Conference on Communications, pp.46.3.1–46.3.5, Alger Press Limited, 1978.
9. J. Kelsey, B. Schneier, D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, proceedings of CRYPTO'96, pp.237–251, Springer Verlag, 1996.
10. J. Kilian, P. Rogaway, *How to Protect Against Exhaustive Key Search*, proceedings of CRYPTO'96, LNCS 1109, pp.252–267, Springer Verlag, 1996.
11. B. Kaliski, M. Robshaw, *Multiple encryption: weighing security and performance*, Dr. Dobb's Journal, pp.123–127, Jan. 1996.
12. L. R. Knudsen, *Cryptanalysis of LOKI91*, proceedings of AUSCRYPT'92, LNCS 718, pp.196–208, Springer Verlag, 1993.
13. M. Luby, C. Rackoff, *How to Construct Pseudorandom Permutations from Pseudorandom Functions*, SIAM Journal of Computing, Vol. 17, pp.373–386, 1988.
14. J. H. Moore, G. J. Simmons, *Cycle Structure of the DES with Weak and Semi-Weak Keys*, proceedings of CRYPTO'86, pp.9–32, Springer Verlag, 1987.
15. B. Preneel, V. Rijmen, A. Bosselears, *Principles and Performance of Cryptographic Algorithms*, Dr. Dobb's Journal, Vol. 23, No. 12, pp.126–131, Miller Freeman, Dec. 1998.
16. P. Rogaway, *The Security of DESX*, RSA Laboratories' CryptoBytes, Summer 1996.
17. B. Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, proceedings of FSE'94, LNCS 809, pp.191–204, Springer Verlag, 1994.
18. C. Shannon, *Communication Theory of Secrecy Systems*, Bell Sys. Tech. J., Vol. 28, pp. 656–715, October 1949. (A declassified report from 1945.)
19. I. A. Zabotin, G. P. Glazkov, V. B. Isaeva, *Cryptographic Protection for Information Processing Systems. Cryptographic Transformation Algorithm*, Government Standard of the USSR, GOST 28147-89, 1989. (Translated by A. Malchik, with editorial and typographic assistance of W. Diffie.)

9. MINING POOLS.

Mining Pools at DES-X: What makes the attraction?

1. Mining Optimization: Mining Pools at DES-X are designed to optimize the mining process, combining the integrated power of many different mining sources. This brings diversity and stability to the mining operation.

2. Fair Profit Sharing: By joining Mining Pools at DES-X, miners have the opportunity to share their fair profits based on their contribution. The profit-sharing process is ensured using fair and transparent mechanisms.

3. Community Building Network: Mining Pools is not only a place to participate in mining but also a platform to promote interaction and information exchange between miners. This helps build a community that is united, capable of learning and interacting.

9.1: Kryptex on DES-X Exchange.

Modern Mining Platform - Kryptex on DES

1. Technological Innovation: Kryptex is more than just a regular cryptocurrency mining software. It offers an innovative experience through the combination of advanced mining algorithms and a user-friendly interface.

2. Performance Optimization: One of the outstanding advantages of Kryptex is its ability to optimize mining performance. This tool helps to make the most of your hardware's capabilities for the best profit.

3. Asset Diversification: Kryptex isn't just focused on a single cryptocurrency. Instead, it supports the mining of various cryptocurrencies, creating an opportunity to diversify profits.

4. Safety and Confidentiality: Kryptex is committed to protecting users' information and assets. The tool uses advanced security measures to ensure a safe and secure mining environment.

Accompanying DES with Kryptex
The partnership between Kryptex and DES Exchange creates a unique opportunity to participate in a growing decentralized trading system. With Kryptex, you are not only mining cryptocurrencies but also contributing to building a futures trading environment.

9.2: OKMINER Joins Trading At DES-X.

Mining Breakthrough With OKMINER on DES-X

1. Outstanding Performance: OKMINER offers outstanding mining performance with advanced and optimized equipment. This combination helps you make the most of your mining power for the best profit.

2. Quality and Reliability: OKMINER has made its mark with the quality and reliability of its mining equipment. This ensures you get a stable and seamless mining experience.

3. Diversification: OKMINER is not just focused on a single cryptocurrency. They support the mining of various cryptocurrencies, helping you to take advantage of the opportunity to diversify your profits.

4. The Future of Decentralized Finance: The combination of OKMINER and DES-X creates a potential decentralized trading environment. Let's join and contribute to building a decentralized and sustainable financial future.

OKMINER has demonstrated quality and innovation in the field of cryptocurrency mining. Join this journey at DES-X Exchange to gain profit and participate in building a world-class decentralized trading environment.

9.3: F2-POOL on DES-X Exchange.

Highlights of F2-POOL on DES-X

1. Diversified Mining: F2-POOL offers diversification in mining operations. With the ability to mine multiple cryptocurrencies, you can optimize profits from different sources and take advantage of opportunities in the diverse market.

2. Optimizing Performance: With professionalism and experience in the field of mining, F2-POOL helps you optimize the performance of your mining equipment to achieve the best profit.

3. Safe and Transparent Transactions: F2-POOL is committed to ensuring safety and transparency in the mining process. This helps you to engage in trading with confidence and peace of mind.

4. Building a Breakthrough Trading Environment: F2-POOL on DES-X is not only part of the decentralized trading environment, but also an important part of creating a diversified financial future. form and develop.

The partnership between F2-POOL and DES-X Exchange gives you an opportunity to join the decentralized finance revolution. You are not only mining cryptocurrencies, but you are also contributing to the creation of a disruptive and diverse trading environment.

10. MINING SOFTWARE.

Discover the Power of Mining Software on DES-X

1. Diversity of Miners: Mining Software at DES-X offers diversity in miners. With compatibility with a wide range of devices and cryptocurrencies, you can get the most out of your mining potential.

2. Optimizing Performance: One of Mining Software's strengths is its ability to optimize the performance of mining equipment. This helps you achieve maximum performance and maximize profits.

3. Safe and Secure Transactions: Mining Software on DES-X is committed to ensuring safety and security in mining activities.

You can count on the transparency and reliability of the mining process on our platform.

10.1: BZMINER on DES-X.

The Power of BZMINER on DES-X

1. Ultimate Performance: BZMINER offers optimal mining

performance with the ability to make the most of the power of the mining equipment. This helps you to get the best profit from your mining operation.

2. Diversity of Assets: BZMINER is not limited to mining a single cryptocurrency. Instead, the tool supports the mining of various cryptocurrencies, providing an opportunity to diversify your profits.

3. Safe and Transparent Transactions: BZMINER is committed to ensuring safety and transparency in the mining process. This helps you to engage in trading with confidence and reliability.

4. Building Diversified Financial Futures: The combination of BZMINER and DES-X creates a diverse and growing mining

environment. Join us to contribute to building the future of decentralized finance full of potential.

Are you ready to discover the power of crypto mining with BZMINER at DES-X? Don't miss the opportunity to join a potential and exciting mining community.

10.2: Wildrig on DES-X.

Wildridge - Discover Wild Inspiration

1. Outstanding Performance: Wildrig offers outstanding mining performance with the ability to optimize the power of mining equipment. This helps you to get the best profit from cryptocurrency mining.

2. Diversity of Assets: Wildrig is not limited to mining a single cryptocurrency. Instead, it supports the mining of various cryptocurrencies, providing an opportunity to diversify your profits.

3. Safe and Reliable Transactions: Wildrig is committed to protecting the safety and reliability of the mining process. This helps you to engage in trading with confidence and peace of mind.

4. Building Diversified Financial Futures: Wildrig contributes to building a diverse and thriving mining environment on DES-X. Join us to take advantage of crypto mining opportunities and contribute to the future of decentralized finance.

There is a wild world waiting for you at Wildrig on the DES-X Exchange. Join this journey to experience the wild inspiration and maximize profits from cryptocurrency mining.

10.3: Rigel - Exploring the Crypto Universe on DES-X

Rigel - When Technology Meets the Universe

1. Super Performance: Rigel offers outstanding mining performance with the ability to optimize the power of mining equipment. This helps you to get maximum profit from cryptocurrency mining.

2. Asset Diversification: Rigel is not limited to mining a single cryptocurrency. Instead, it supports the mining of various cryptocurrencies, providing an opportunity to diversify your profits.

3. Safe and Reliable: Rigel is committed to ensuring safety and reliability during mining. This helps you to engage in trading with confidence and peace of mind.

4. Building Diversified Financial Futures: Rigel contributes to building a diverse and thriving mining environment at DES-X. Join us to take advantage of crypto mining opportunities and contribute to the future of decentralized finance.

The crypto universe awaits you at Rigel on the DES-X Exchange. Join this journey to experience the fusion of technology and explore the crypto universe.

10.4: LOLMINER - Bringing Fun and Performance to the DES-X.

LOLMINER - Bringing Fun to Mining

1. Outstanding Performance: LOLMINER offers outstanding mining performance with the ability to optimize the power of mining equipment. This helps you to get maximum profit from cryptocurrency mining.

2. Asset Diversification: LOLMINER is not limited to mining a single cryptocurrency. Instead, it supports the mining of various

cryptocurrencies, helping you to take advantage of the opportunity to diversify your profits.

3. Safe and Reliable Transactions: LOLMINER is committed to protecting the safety and reliability of the mining process. This helps you to engage in trading with confidence and peace of mind.

4. Building Diversified Financial Futures: The combination of LOLMINER and DES-X creates a diverse and growing mining environment. Join us to take advantage of crypto mining opportunities and contribute to the future of decentralized finance.

With LOLMINER on the DES-X Exchange, you are not only mining cryptocurrencies but also embarking on a fun and challenging journey. Get ready to explore the new world of cryptocurrency mining with a combination of fun and performance.

11. Benefits of DES-X exchange participants.

DES-X Exchange is not only a place to trade cryptocurrencies, but also a diverse and potential community. In this article, we will go over the outstanding benefits that participants can experience

when participating in DES-X's unique decentralized trading environment.

11.1: Diversified Profits

Participants of the DES-X exchange have the opportunity to take advantage of profit diversification through cryptocurrency mining, trading, and other financial activities. Depending on each person's goals and knowledge, profits can come from mining, investing, staking, or yield farming.

11.2: Effective Exploitation

DES-X exchange cooperates with leading mining providers like OKMINER, F2-POOL, Wildridge, and many more. This helps participants make the most of the performance of their mining devices, achieving optimal profits and efficient mining.

11.3: Decentralized Finance

By participating in DES-X, participants contribute to the development of the decentralized financial environment. They have the opportunity to participate in the construction of a

decentralized financial future and create a safe and transparent trading environment.

11.4: Safe Trading Environment.

DES-X exchange is committed to protecting safety and ensuring transparency in the trading and mining process. Participants can be assured of the security of their assets and personal information.

11.5: Learning Opportunities.

Participants of the DES-X exchange have the opportunity to learn from experts in the field of cryptocurrencies and decentralized finance. They can access instructional materials, and learn how to mine, trade, and manage assets intelligently and efficiently.

11.6: Diverse Community

DES-X is a diverse community that attracts cryptocurrency and decentralized finance enthusiasts from around the world. Participants have the opportunity to network, learn and share experiences with people with similar interests.

11.7: Growth Potential

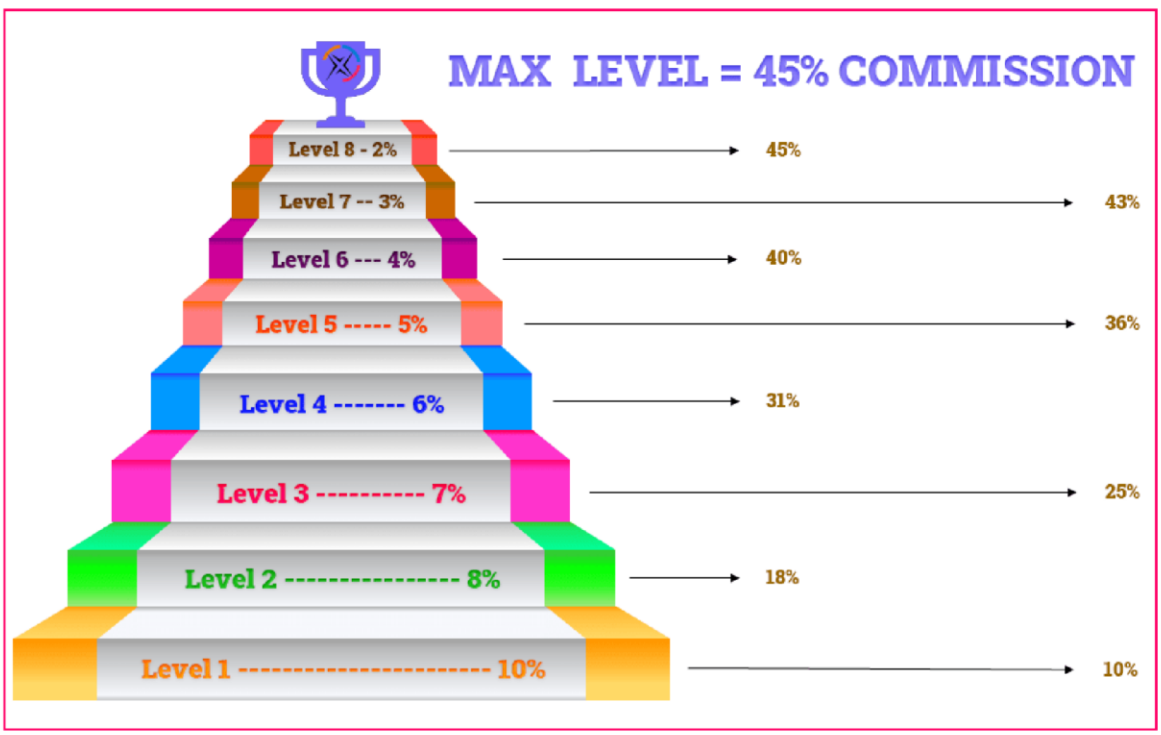
With the continuous development of cryptocurrencies and decentralized finance, participants of the DES-X exchange have the opportunity to enter an area with strong growth potential in the future.

12. Invite Friends Program.

Reward yourself and help your friends! Internet income is unlimited, with an advanced muli-level bonus mechanism DESX Bonuses are distributed instantly and processed fully automatically by a smart contract. The multi-level bounty mechanism on the blockchain is consistent with infallible logic.

Referral rewards

With a commission rate of 10% for each invite from level 1.
With a commission rate of 08% for each invite from level 2.
With a commission rate of 07% for each invite from level 3.
With a commission rate of 06% for each invite from level 4.
With a commission rate of 05% for each invite from level 5.
With a commission rate of 04% for each invite from level 6.
With a commission rate of 03% for each invite from level 7.
With a commission rate of 02% for each invite from level 8.



NOTICE AND DISCLAIMER

PLEASE READ THE ENTIRETY OF THIS "NOTICE AND DISCLAIMER" SECTION CAREFULLY. NOTHING HEREIN CONSTITUTES LEGAL, FINANCIAL, BUSINESS, OR TAX ADVICE AND YOU SHOULD CONSULT YOUR OWN LEGAL, FINANCIAL, TAX, OR OTHER PROFESSIONAL ADVISOR(S) BEFORE ENGAGING IN ANY ACTIVITY IN CONNECTION HEREWITH. NOR ANY SERVICE PROVIDER SHALL BE LIABLE FOR ANY KIND OF DIRECT OR INDIRECT DAMAGE OR LOSS WHATSOEVER WHICH YOU MAY SUFFER IN CONNECTION WITH ACCESSING THIS WHITEPAPER, THE WEBSITE AT https://des-x.io/(THE WEBSITE) OR ANY OTHER WEBSITES OR MATERIALS PUBLISHED BY THE FOUNDATION.

Nature of the Whitepaper: The Whitepaper and the Website are intended for general informational purposes only and do not constitute a prospectus, an offer document, an offer of securities, a solicitation for investment, or any offer to sell any product, item, or asset (whether digital or otherwise). The information herein may not be exhaustive and does not imply any element of a contractual relationship. There is no assurance as to the accuracy or completeness of such information and no representation, warranty, or undertaking is or purported to be provided as to the accuracy or completeness of such information. Where the Whitepaper or the Website includes information that has been obtained from third-party sources, the Foundation, the Distributor, their respective affiliates, and/or the DESX team have not independently verified the accuracy or completion of such information. Further, you acknowledge that circumstances may change and that the Whitepaper or the Website may become outdated as a result; and neither the Foundation nor the Distributor

is under any obligation to update or correct this document in connection therewith.