КРИПТОНИТ

Format Preserving Encryption

A survey

Tsaregorodtsev Kirill

2nd June 2021

Table of contents

Statement of the problem

What is FPE?

Definition (FPE)
An encryption algorithm with the following property: the resulting ciphertext format must be the same as the format of plaintext.

What is FPE?

Definition (FPE)
An encryption algorithm with the following property: the resulting ciphertext format must be the same as the format of plaintext.

Definition (FPE, more formally)
A pair of algorithms:

$$E,\ D : \mathtt{Keys} \times \mathtt{Twk} \times \mathtt{Dom} \rightarrow \mathtt{Dom},$$

What is FPE?

Definition (FPE)
An encryption algorithm with the following property: the resulting ciphertext format must be the same as the format of plaintext.

Definition (FPE, more formally)
A pair of algorithms:

$$E,\ D : \mathtt{Keys} \times \mathtt{Twk} \times \mathtt{Dom} \to \mathtt{Dom},$$

such that

$$D_k^t(E_k^t(m)) = m,$$

What is FPE?

Definition (FPE)
An encryption algorithm with the following property: the resulting ciphertext format must be the same as the format of plaintext.

Definition (FPE, more formally)
A pair of algorithms:

$$E, D : \mathbf{Keys} \times \mathbf{Twk} \times \mathbf{Dom} \to \mathbf{Dom},$$

such that

$$D_k^t(E_k^t(m)) = m,$$

where:

$$\begin{aligned} t \in \mathbf{Twk} &\quad \text{is a tweak (a block cipher parameter),} \\ k \in \mathbf{Keys} &\quad \text{is a key,} \\ m \in \mathbf{Dom} &\quad \text{is a message.} \end{aligned}$$

What is FPE?

Definition (FPE)
An encryption algorithm with the following property: the resulting ciphertext format must be the same as the format of plaintext.

Definition (FPE, more formally)
A pair of algorithms:

$$E,\ D : \mathbf{Keys} \times \mathbf{Twk} \times \mathbf{Dom} \to \mathbf{Dom},$$

such that

$$D_k^t(E_k^t(m)) = m,$$

where:

$$\begin{aligned} t \in \mathbf{Twk} &\ \text{ is a tweak (a block cipher parameter),} \\ k \in \mathbf{Keys} &\ \text{ is a key,} \\ m \in \mathbf{Dom} &\ \text{ is a message.} \end{aligned}$$

Tweak space might be empty: $\mathbf{Twk} = \varnothing$ (more on that later).

Why do we need FPE?

Two examples:

Why do we need FPE?

Two examples:

1. Database structure may be incompatible with encrypted messages $\Rightarrow$ restructure the database or use FPE;

Why do we need FPE?

Two examples:

1. Database structure may be incompatible with encrypted messages ⇒ restructure the database or use FPE;
2. Some applications may require the data to be in a pre-defined format ⇒ rewrite an application from scratch or use FPE;

Why do we need FPE?

Two examples:

1. Database structure may be incompatible with encrypted messages ⇒ restructure the database or use FPE;
2. Some applications may require the data to be in a pre-defined format ⇒ rewrite an application from scratch or use FPE;

Block cipher is not enough: it acts as a permutation on the fixed length binary strings (for instance, $\{0,1\}^{128}$ for «Kuznyechik»).

Why do we need FPE?

Two examples:

1. Database structure may be incompatible with encrypted messages $\Rightarrow$ restructure the database or use FPE;
2. Some applications may require the data to be in a pre-defined format $\Rightarrow$ rewrite an application from scratch or use FPE;

Block cipher is not enough: it acts as a permutation on the fixed length binary strings (for instance, $\{0,1\}^{128}$ for «Kuznyechik»).

Even if $\mathbf{Dom} \subseteq \{0,1\}^n$, the result $m \to E_k(m) \notin \mathbf{Dom}$ with high probability (due to its relatively small size in the real-world situations and applications).

Message space from the real world

Example (SNILS)

If we encrypt 9-digit individual insurance account number (SNILS), the result must be 9-digit ciphertext, i.e.

$$\mathtt{Dom} = \{0, 1, \ldots, 9\}^9$$

Message space from the real world

Example (SNILS)
If we encrypt 9-digit individual insurance account number (SNILS), the result must be 9-digit ciphertext, i.e.

$$\mathbf{Dom} = \{0, 1, \dots, 9\}^9$$

Example (CCN)
CCN consist of the following numbers:

- **6 digits** — bank number,
- **6 digits** — account number,
- **3 digits** — checksum,

and all digits, except for account number (i.e., 9 out of 15), are publicly available. In this case:

$$\mathbf{Dom} = \{0, \dots 9\}^6.$$

Why do we use tweaks?

- Usual block cipher: big codebook (a set of pairs $(m, E_k(m))$), hence hard to collect $\Rightarrow$ ignore exhaustive attacks;

Why do we use tweaks?

- Usual block cipher: big codebook (a set of pairs $(m, E_k(m))$), hence hard to collect $\Rightarrow$ ignore exhaustive attacks;
- Small domain size (and codebook) for FPE $\Rightarrow$ severe practical threat (dictionary attack);

Why do we use tweaks?

- Usual block cipher: big codebook (a set of pairs $(m, E_k(m))$), hence hard to collect $\Rightarrow$ ignore exhaustive attacks;
- Small domain size (and codebook) for FPE $\Rightarrow$ severe practical threat (dictionary attack);

Example (Dictionary attack)

CCNs from various banks can have the same account number $\Rightarrow$ matching ciphertext blocks correspond to matching plaintext.

Why do we use tweaks?

- Usual block cipher: big codebook (a set of pairs $(m, E_k(m))$), hence hard to collect $\Rightarrow$ ignore exhaustive attacks;
- Small domain size (and codebook) for FPE $\Rightarrow$ severe practical threat (dictionary attack);

Example (Dictionary attack)

CCNs from various banks can have the same account number $\Rightarrow$ matching ciphertext blocks correspond to matching plaintext.

Bank number	Account number	Checksum
012345	$E_k(000111)$	123
	$\updownarrow$ **same**	
987654	$E_k(000111)$	456

Why do we use tweaks?

- Usual block cipher: big codebook (a set of pairs $(m, E_k(m))$), hence hard to collect $\Rightarrow$ ignore exhaustive attacks;
- Small domain size (and codebook) for FPE $\Rightarrow$ severe practical threat (dictionary attack);

Example (Dictionary attack)

CCNs from various banks can have the same account number $\Rightarrow$ matching ciphertext blocks correspond to matching plaintext.

Bank number	Account number	Checksum
012345	$E_k(000111)$	123
	$\updownarrow$ same	
987654	$E_k(000111)$	456

The main goal of the (non-secret) tweak is to expand the set of possible permutations;

What is a good FPE algorithm?

What is a good FPE algorithm?

- The cipher must look like a random permutation on the given (usually small) domain **Dom**.

What is a good FPE algorithm?

- The cipher must look like a random permutation on the given (usually small) domain **Dom**.
- An adversary can usually obtain all ciphertexts of all points in the domain due to its small size (exhaustive search), hence the tweak.

What is a good FPE algorithm?

- The cipher must look like a random permutation on the given (usually small) domain **Dom**.
- An adversary can usually obtain all ciphertexts of all points in the domain due to its small size (exhaustive search), hence the tweak.
- The algorithm must be efficient.

What is a good FPE algorithm?

- The cipher must look like a random permutation on the given (usually small) domain **Dom**.
- An adversary can usually obtain all ciphertexts of all points in the domain due to its small size (exhaustive search), hence the tweak.
- The algorithm must be efficient.
- It is desirable to use existing well-studied primitives and principles: block ciphers, Feistel networks.

What is a good FPE algorithm formally?

- Lot of ways to formalize what we want from FPE algorithm;

What is a good FPE algorithm formally?

- Lot of ways to formalize what we want from FPE algorithm;
- One of the possible formalizations: «weakly dependent» permutations for different chosen $t \in \mathtt{Twk}$ (tweakable pseudorandom permutation);

What is a good FPE algorithm formally?

- Lot of ways to formalize what we want from FPE algorithm;
- One of the possible formalizations: «weakly dependent» permutations for different chosen $t \in \mathbf{Twk}$ (tweakable pseudorandom permutation);

Algorithm 5 Experiment Left

```
function INIT
    for t ∈ Twk do
        π^t ←R Perm(Dom)
function O(t, m)
    return π^t(m)
```

What is a good FPE algorithm formally?

- Lot of ways to formalize what we want from FPE algorithm;
- One of the possible formalizations: «weakly dependent» permutations for different chosen $t \in \mathbf{Twk}$ (tweakable pseudorandom permutation);

Algorithm 7 Experiment Left

```
function INIT
    for t ∈ Twk do
        π^t ←R Perm(Dom)
function O(t, m)
    return π^t(m)
```

Algorithm 8 Experiment Right

```
function INIT
    k ←$ Keys
function O(t, m)
    return E_k^t(m)
```

Formalization

Let $Adv_E^{TPRP}(\mathcal{A})$ be the advantage of the adversary $\mathcal{A}$ in the distinguishing attack, i.e.:

$$Adv_E^{TPRP}(\mathcal{A}) = \mathbb{P}[Right(\mathcal{A}) \to 1] - \mathbb{P}[Left(\mathcal{A}) \to 1].$$

Formalization

Let $Adv_E^{TPRP}(\mathcal{A})$ be the advantage of the adversary $\mathcal{A}$ in the distinguishing attack, i.e.:

$$Adv_E^{TPRP}(\mathcal{A}) = \mathbb{P}[Right(\mathcal{A}) \to 1] - \mathbb{P}[Left(\mathcal{A}) \to 1].$$

The probability $\mathbb{P}[\cdot]$:

1. is taken over random choise of permutations $\pi^t \leftarrow^R Perm(\mathtt{Dom})$ for different t and random coins of $\mathcal{A}$ (if any) — in case of Left experiment;
2. is taken over random choise of key $k \leftarrow^\$ \mathtt{Keys}$ and random coins of $\mathcal{A}$ (if any) — in case of Right experiment;

Formalization

Let $Adv_E^{TPRP}(\mathcal{A})$ be the advantage of the adversary $\mathcal{A}$ in the distinguishing attack, i.e.:

$$Adv_E^{TPRP}(\mathcal{A}) = \mathbb{P}[Right(\mathcal{A}) \to 1] - \mathbb{P}[Left(\mathcal{A}) \to 1].$$

The probability $\mathbb{P}[\cdot]$:

1. is taken over random choise of permutations $\pi^t \xleftarrow{R} Perm(\mathbf{Dom})$ for different t and random coins of $\mathcal{A}$ (if any) — in case of Left experiment;
2. is taken over random choise of key $k \xleftarrow{\$} \mathbf{Keys}$ and random coins of $\mathcal{A}$ (if any) — in case of Right experiment;

The algorithm is «good» if the maximal advantage is «small».

Proposed solutions

Overview of solutions

- NIST standardization: FF1-FF3, FEA-2;

Overview of solutions

- NIST standardization: FF1-FF3, FEA-2;
- General techniques: cycle walking;

Overview of solutions

- NIST standardization: FF1-FF3, FEA-2;
- General techniques: cycle walking;
- Small domain solutions: prefix encryption, various shuffling techniques;

Overview of solutions

- NIST standardization: FF1-FF3, FEA-2;
- General techniques: cycle walking;
- Small domain solutions: prefix encryption, various shuffling techniques;
- Primitive layer solutions (SPF);

FF1, FF3

- Semi-balanced Feistel network over the group $\mathtt{Dom} = \mathbb{Z}_M \times \mathbb{Z}_N$, where $M \approx N$

FF1, FF3

- Semi-balanced Feistel network over the group $\mathtt{Dom} = \mathbb{Z}_M \times \mathbb{Z}_N$, where $M \approx N$
- The algorithm takes the key $k \in \mathtt{Keys}$, the element to be encrypted $(A, B) \in \mathtt{Dom}$, and the tweak $t \in \mathtt{Twk}$ (usually tweak space $\mathtt{Twk}$ is of the form $\{0,1\}^{tlen}$).

FF1, FF3

- Semi-balanced Feistel network over the group $\mathbf{Dom} = \mathbb{Z}_M \times \mathbb{Z}_N$, where $M \approx N$
- The algorithm takes the key $k \in \mathbf{Keys}$, the element to be encrypted $(A, B) \in \mathbf{Dom}$, and the tweak $t \in \mathbf{Twk}$ (usually tweak space $\mathbf{Twk}$ is of the form $\{0,1\}^{tlen}$).
- One round is of the form:

$$(A, B) \to (B, A \boxplus Q),$$

where $Q = PRF_k(B, t, i, params)$, i is the round number, $params$ is some (non-secret) information, PRF is a pseudorandom function

FF1, FF3

- Semi-balanced Feistel network over the group **Dom** $= \mathbb{Z}_M \times \mathbb{Z}_N$, where $M \approx N$
- The algorithm takes the key $k \in$ **Keys**, the element to be encrypted $(A, B) \in$ **Dom**, and the tweak $t \in$ **Twk** (usually tweak space **Twk** is of the form $\{0,1\}^{tlen}$).
- One round is of the form:

$$(A, B) \to (B, A \boxplus Q),$$

where $Q = PRF_k(B, t, i, params)$, i is the round number, $params$ is some (non-secret) information, PRF is a pseudorandom function
- Suggestion: 10 rounds of Feistel network are enough for FF1 security and 8 rounds for FF3.

FF1, FF3: PROs and CONs

1. Patarin articles on the (classical) Feistel networks;

1. Patarin articles on the (classical) Feistel networks;
2. Papers on the Feistel networks over groups $\mathbb{Z}_M \times \mathbb{Z}_N$;

FF1, FF3: PROs and CONs

1. Patarin articles on the (classical) Feistel networks;
2. Papers on the Feistel networks over groups $\mathbb{Z}_M \times \mathbb{Z}_N$;
3. But: no provable security;

FF1, FF3: PROs and CONs

1. Patarin articles on the (classical) Feistel networks;
2. Papers on the Feistel networks over groups $\mathbb{Z}_M \times \mathbb{Z}_N$;
3. But: no provable security;
4. But: bad tweak mixing in FF3 $\Rightarrow$ some specific attacks on FF3;

FF2

Given parameters: $k \in \texttt{Keys}, t \in \texttt{Twk}, m \in \texttt{Dom}$.

FF2

Given parameters: $k \in \mathtt{Keys}, t \in \mathtt{Twk}, m \in \mathtt{Dom}$.

Two-step procedure:

FF2

Given parameters: $k \in \texttt{Keys}, t \in \texttt{Twk}, m \in \texttt{Dom}$.

Two-step procedure:

1. Derive the secret key for the given tweak:

$$sk = E_k(t) \in \{0, 1\}^{128};$$

FF2

Given parameters: $k \in \mathtt{Keys}, t \in \mathtt{Twk}, m \in \mathtt{Dom}$.

Two-step procedure:

1. Derive the secret key for the given tweak:

$$sk = E_k(t) \in \{0, 1\}^{128};$$

2. Encrypt the message with obtained key

$$c = Feistel_{sk}(m);$$

FF2

Given parameters: $k \in \mathtt{Keys}, t \in \mathtt{Twk}, m \in \mathtt{Dom}$.

Two-step procedure:

1. Derive the secret key for the given tweak:

$$sk = E_k(t) \in \{0,1\}^{128};$$

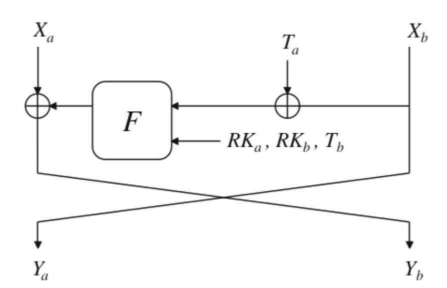
2. Encrypt the message with obtained key

$$c = Feistel_{sk}(m);$$

Main flaw: the key length $|sk| = 128$ is too short to guarantee the strong security bound.

FEA-2

One round of the proposed scheme:



$X_a \,||\, X_b$ — left and right blocks of the message;
$T_a \,||\, T_b$ — left and right blocks of the tweak;
$RK_a \,||\, RK_b$ — left and right blocks of the round key;

It is assumed that $|T_a| = |X_b|, |T_a| + |T_b| = 128 = |RK_a| = |RK_b|$).

FEA-2 features

Some remarkable features of the algorithm:

FEA-2 features

Some remarkable features of the algorithm:

1. Tweak is embedded at the primitive layer, i.e., is used in each encryption round.

FEA-2 features

Some remarkable features of the algorithm:

1. Tweak is embedded at the primitive layer, i.e., is used in each encryption round.
2. FEA-2 is able to encrypt messages of various lenghts, $\mathbf{Dom} = \{0, 1\}^n$, where $n \in \{8, 9, \dots 128\}$.

FEA-2 features

Some remarkable features of the algorithm:

1. Tweak is embedded at the primitive layer, i.e., is used in each encryption round.
2. FEA-2 is able to encrypt messages of various lenghts, $\mathtt{Dom} = \{0,1\}^n$, where $n \in \{8, 9, \dots 128\}$.
3. The number of rounds depends on the block size and starts with 18.

FEA-2 features

Some remarkable features of the algorithm:

1. Tweak is embedded at the primitive layer, i.e., is used in each encryption round.
2. FEA-2 is able to encrypt messages of various lenghts, **Dom** $= \{0, 1\}^n$, where $n \in \{8, 9, \dots 128\}$.
3. The number of rounds depends on the block size and starts with 18.
4. The key lenght is not fixed: $klen \in \{128, 192, 256\}$.

FEA-2 features

Some remarkable features of the algorithm:

1. Tweak is embedded at the primitive layer, i.e., is used in each encryption round.
2. FEA-2 is able to encrypt messages of various lenghts, $\mathbf{Dom} = \{0,1\}^n$, where $n \in \{8, 9, \dots 128\}$.
3. The number of rounds depends on the block size and starts with 18.
4. The key lenght is not fixed: $klen \in \{128, 192, 256\}$.
5. Encryption on the domain $\mathbf{Dom} = \{1, \dots, N\}$ is done via embedding $\mathbf{Dom} \subseteq \{0,1\}^n$ combined with the cycle walking idea (more on that later).

FEA-2 features

Some remarkable features of the algorithm:

1. Tweak is embedded at the primitive layer, i.e., is used in each encryption round.
2. FEA-2 is able to encrypt messages of various lenghts, $\mathbf{Dom} = \{0, 1\}^n$, where $n \in \{8, 9, \ldots 128\}$.
3. The number of rounds depends on the block size and starts with 18.
4. The key lenght is not fixed: $klen \in \{128, 192, 256\}$.
5. Encryption on the domain $\mathbf{Dom} = \{1, \ldots, N\}$ is done via embedding $\mathbf{Dom} \subseteq \{0, 1\}^n$ combined with the cycle walking idea (more on that later).
6. Linear and differential analysis, related-key attacks and threats, specific to Feistel networks over small domains, were investigated. It was claimed that for the domains of size greater than 2^8 the proposed attacks require at least 2^{64} encryptions on different parameters $t \in \mathbf{Twk}$.

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\mathbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\textbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

1. For $m \in \textbf{Dom}$ compute $c \leftarrow E_k(m)$.

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\mathbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

1. For $m \in$ **Dom** compute $c \leftarrow E_k(m)$.
2. If $c \in$ **Dom**, then m maps to c.

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\mathbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

1. For $m \in$ **Dom** compute $c \leftarrow E_k(m)$.
2. If $c \in$ **Dom**, then m maps to c.
3. If $c \notin$ **Dom**, then $c \leftarrow E_k(c)$ and go to step 2.

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\mathbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

1. For $m \in \mathbf{Dom}$ compute $c \leftarrow E_k(m)$.
2. If $c \in \mathbf{Dom}$, then m maps to c.
3. If $c \notin \mathbf{Dom}$, then $c \leftarrow E_k(c)$ and go to step 2.

- The algorithm is provably secure;

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\mathbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

1. For $m \in \mathbf{Dom}$ compute $c \leftarrow E_k(m)$.
2. If $c \in \mathbf{Dom}$, then m maps to c.
3. If $c \notin \mathbf{Dom}$, then $c \leftarrow E_k(c)$ and go to step 2.

- The algorithm is provably secure;
- The expected number of encryption operations (before one obtains $c \in \mathbf{Dom}$) is determined by the quantity $\frac{2^n}{|\mathbf{Dom}|}$;

Cycle walking

If **Dom** is «close» to $\{0,1\}^n$ (i.e. $\frac{|\mathbf{Dom}|}{2^n} \approx 1$) for some standard block size n, then the following approach works:

1. For $m \in \mathbf{Dom}$ compute $c \leftarrow E_k(m)$.
2. If $c \in \mathbf{Dom}$, then m maps to c.
3. If $c \notin \mathbf{Dom}$, then $c \leftarrow E_k(c)$ and go to step 2.

- The algorithm is provably secure;
- The expected number of encryption operations (before one obtains $c \in \mathbf{Dom}$) is determined by the quantity $\frac{2^n}{|\mathbf{Dom}|}$;
- Side-channel attacks (time) does not give an information to the adversary;

Prefix encryption

If the domain **Dom** is small enough, then we can use the following idea:

Prefix encryption

If the domain **Dom** is small enough, then we can use the following idea:

1. Compute the number list:

$$S = (E_k(0), \dots, E_k(N-1));$$

Prefix encryption

If the domain **Dom** is small enough, then we can use the following idea:

1. Compute the number list:

$$S = (E_k(0), \dots, E_k(N-1));$$

2. To encrypt the message $m \in \mathbb{Z}_N$ map m to the position of $E_k(m)$ in the sorted list.

Prefix encryption

If the domain **Dom** is small enough, then we can use the following idea:

1. Compute the number list:

$$S = (E_k(0), \dots, E_k(N-1));$$

2. To encrypt the message $m \in \mathbb{Z}_N$ map m to the position of $E_k(m)$ in the sorted list.

The method is provably secure but requires $O(N)$ encryption operations at the initial step and $O(N)$ memory to store the table.

Attacks on the solutions

Attack on FF2

Assume that the adversary has n ciphertexts of the form:

$$c_j = Feistel_{E_k(t_j)}(m).$$

Attack on FF2

Assume that the adversary has n ciphertexts of the form:

$$c_j = Feistel_{E_k(t_j)}(m).$$

Then he can try different keys $sk_j \in \{0,1\}^{128}$ and obtain

$$c'_j = Feistel_{sk_j}(m).$$

Attack on FF2

Assume that the adversary has n ciphertexts of the form:

$$c_j = Feistel_{E_k(t_j)}(m).$$

Then he can try different keys $sk_j \in \{0,1\}^{128}$ and obtain

$$c'_j = Feistel_{sk_j}(m).$$

If $c'_j = c_i$, then (with high probability) $sk_j = E_k(t_j)$, i.e., he can recover the derived key for the given tweak t_j without knowing the value of master-key k.

Attack on FF2

Assume that the adversary has n ciphertexts of the form:

$$c_j = Feistel_{E_k(t_j)}(m).$$

Then he can try different keys $sk_j \in \{0,1\}^{128}$ and obtain

$$c'_j = Feistel_{sk_j}(m).$$

If $c'_j = c_i$, then (with high probability) $sk_j = E_k(t_j)$, i.e., he can recover the derived key for the given tweak t_j without knowing the value of master-key k.

If the number of ciphertexts $n = 2^u$, then the collision is expected to occur after 2^{128-u} steps.

Attacks on FF1, FF3

Two types of attacks:

Attacks on FF1, FF3

Two types of attacks:

1. Exploit the wrong design of tweak mixing (specific for FF3). In these attacks, the adversary adaptively chooses plaintexts to be encrypted on two selected $t_1, t_2 \in \texttt{Twk}$.

Attacks on FF1, FF3

Two types of attacks:

1. Exploit the wrong design of tweak mixing (specific for FF3). In these attacks, the adversary adaptively chooses plaintexts to be encrypted on two selected $t_1, t_2 \in \mathbf{Twk}$.
2. Intrinsic feature of Feistel network over small domains: the proposed number of rounds is not enough to hide the plaintext statistics (a slight bias after one round of Feistel network, which can be amplified using different tweaks $t \in \mathbf{Twk}$ for the same message).

Attacks on FF1, FF3: remarks

Some remarks:

Attacks on FF1, FF3: remarks

Some remarks:

- The number of required texts formally exceeds the domain size;

Attacks on FF1, FF3: remarks

Some remarks:

- The number of required texts formally exceeds the domain size;
- In fact only a minimal (even constant) number of texts are required for each $t \in \texttt{Twk}$.

Attacks on FF1, FF3: remarks

Some remarks:

- The number of required texts formally exceeds the domain size;
- In fact only a minimal (even constant) number of texts are required for each $t \in \texttt{Twk}$.
- This fact was not reflected in the original security model. All the proofs were obtained in a weaker model, in which the adversary cannot make the number of requests to the oracle that exceeds the domain size.

Notation

The following notation is used:

- n — bitsize of one half of the message $m \in \mathbf{Dom}$, i.e. $\mathbf{Dom} = \{0, 1\}^{2n}$;
- N — number of different halves of the message, i.e. $N = 2^n$;
- r — number of rounds in Feistel network;
- q — number of oracle queries;
- t — time complexity (in parrots);

Short summary of attacks: 1

Year: 2004

Threat: distinguisher, **generic Feistel network**

Resources: $q_t = N^{r-2}$ encryptions queries on different $t \in \texttt{Twk}$, two messages per tweak ($q_e = 2$), time complexity $t \approx q_t q_e$

Comments: attack distinguishes Feistel network output from a random string

Short summary of attacks: 2

Year: 2016

Threat: message recovery, **generic Feistel network**

Resources: $q_t = \mathcal{O}(n \cdot N^{r-2})$ encryptions queries on different $t \in$ **Twk**, 3 messages per tweak ($q_e = 3$), time complexity $t \approx q_t$

Comments:

1. The adversary knows ciphertexts of three different messages (x, x', x^*) under tweaks $t_1, \dots, t_q$, and recovers the message x.
2. The message x' is fully known to the adversary but unrelated to x.
3. x^* and x share a common right side; only the left side of x^* is known to the adversary.
4. The attack is not adaptive; only the knowledge of plaintexts is required.

Short summary of attacks: 3

Year: 2017

Threat: Entire codebook recovery for t_1, t_2 for **FF3**.

Resources: $q_e = \mathcal{O}(N^{\frac{11}{6}})$ encryption queries on two tweaks $t_1, t_2 \in \textbf{Twk}$ $(q_t = 2)$; time complexity $t = \mathcal{O}(N^5)$

Comments:

1. The adaptive choice of messages is required.
2. We assume that the adversary can control the choice of $t \in \textbf{Twk}$. The attack does not work if the adversary does not have complete control over t. Partial truncation of the tweak can be applied to prevent the threat.

Short summary of attacks: 4

Year: 2018

Threat: Recovery of multiple messages $m_1, \dots, m_p$ **generic Feistel network**

Resources: $q_t = \mathcal{O}(N^{r-4}(n \cdot N + p))$ different tweaks, number of plaintexts per tweak: $q_e = \mathcal{O}(n \cdot N)$, time complexity $t = \mathcal{O}(n \cdot N^{r-2}(n + p))$

Comments:

1. The attack is not adaptive; only the knowledge of plaintexts is required.
2. It is assumed that the adversary knows ciphertexts for τ known plaintexts $x_1, \dots x_\tau$ and for p messages (plaintexts) under attack $m_1, \dots m_p$ for q different tweaks.
3. It is assumed that right halves of $x_1, \dots, x_\tau$ comprise all possible right halves of messages.
4. The correlation between $x_1, \dots, x_\tau$ and $m_1, \dots m_p$ is not required.

Short summary of attacks: 5

Year: 2019

Threat: Entire codebook recovery for t_1, t_2 for **FF3**.

Resources: $q_e = \mathcal{O}(N^{\frac{11}{6}})$ encryption queries on two tweaks $t_1, t_2 \in \mathsf{Twk}, q_t = 2$; time complexity $t = \mathcal{O}(N^{\frac{17}{6}})$

Comments:

1. The attack is the strengthened version of the first attack on FF3.
2. The adaptive choice of messages is required.
3. The attack does not work if the adversary cannot obtain full control over t.

Recent attack on Feistel Networks (2020)

Algorithm	Resourses	Threat
FF1, $klen = 128, r = 10$	$q = 2^{60}, t = 2^{70}$	Distinguishing attack
FF3-1, $klen = 128, r = 8$	$q = 2^{80}, t = 2^{100}$	Distinguishing attack
FEA-2, $klen = 128, r = 18$	$q = 2^{80}, t = 2^{84}$	Distinguishing attack
FEA-2, $klen = 256, r = 24$	$q = 2^{80}, t = 2^{84}$	Distinguishing attack
FEA-1, $klen = 192, r = 14$	$q = 2^{36}, t = 2^{136}$	Key recovery
FEA-1, $klen = 256, r = 16$	$q = 2^{48}, t = 2^{136}$	Key recovery
Generic Feistel network	$q = 2^{n(r-4)}, t = 2^{n(r-3)}$	Distinguishing attack

State of the art

State of the art

Large domains: OK (disk encryption);

State of the art

Large domains: OK (disk encryption);

Tiny domains: OK (provably secure shuffling methods);

State of the art

Large domains: OK (disk encryption);

Tiny domains: OK (provably secure shuffling methods);

Small domains: no standardized provably secure solution so far, all NIST and ISO candidates are (theoretically) broken.

Quasigroup based FPE

Quasigroup

Definition (Quasigroup)
A set Q with a binary operation on it:

$$\circ : Q \times Q \to Q,$$

which obeys the following property:

Definition (Quasigroup)
A set Q with a binary operation on it:

$$\circ : Q \times Q \to Q,$$

which obeys the following property: for each $a, b \in Q$ there exist unique $x, y \in Q$ such that:

$$a \circ x = b, \qquad y \circ a = b.$$

Quasigroup

Definition (Quasigroup)
A set Q with a binary operation on it:

$$\circ : Q \times Q \to Q,$$

which obeys the following property: for each $a, b \in Q$ there exist unique $x, y \in Q$ such that:

$$a \circ x = b, \qquad y \circ a = b.$$

Equivalently, operations of left and right multiplication

$$L_a : Q \to Q,\ L_a(x) = a \circ x$$

$$R_a : Q \to Q,\ R_a(y) = y \circ a$$

are bijections on Q.

Pseudorandom permutations on Q

We want to measure how close the composition of quasigroup operations (for instance, left multiplications) to the random permutation on Q.

Algorithm 9 Experiment Left

function INIT(λ)
    $\pi \leftarrow^{R} Perm(Q)$
function $\mathcal{O}(m)$
    return $\pi(m)$

Pseudorandom permutations on Q

We want to measure how close the composition of quasigroup operations (for instance, left multiplications) to the random permutation on Q.

Algorithm 11 Experiment Left

```
function INIT(λ)
    π ←R Perm(Q)
function O(m)
    return π(m)
```

$\textbf{function}\ \textsc{Init}(\lambda)$

$\quad \pi \xleftarrow{R} Perm(Q)$

$\textbf{function}\ \mathcal{O}(m)$

$\quad \textbf{return}\ \pi(m)$

Algorithm 12 Experiment Right

$\textbf{function}\ \textsc{Init}(\lambda)$

$\quad k_1, \ldots, k_\lambda \xleftarrow{R} Q$

$\textbf{function}\ \mathcal{O}(m)$

$\quad \textbf{return}\ k_1 \circ (k_2 \circ (\ldots (k_\lambda \circ m) \ldots)$

Adversary

An adversary $\mathcal{A}$ tries to distinguish between random and «structured» permutation.

$$Adv_Q^{PRP}(\mathcal{A}) = \mathbb{P}[Right(\mathcal{A}) \to 1] - \mathbb{P}[Left(\mathcal{A}) \to 1].$$

Adversary

An adversary $\mathcal{A}$ tries to distinguish between random and «structured» permutation.

$$Adv_Q^{PRP}(\mathcal{A}) = \mathbb{P}[Right(\mathcal{A}) \to 1] - \mathbb{P}[Left(\mathcal{A}) \to 1].$$

$$InSec_Q^{PRP}(t, q) = \max_{\mathcal{A} \in A(t,q)} (Adv_Q^{PRP}(\mathcal{A})),$$

Adversary

An adversary $\mathcal{A}$ tries to distinguish between random and «structured» permutation.

$$Adv_Q^{PRP}(\mathcal{A}) = \mathbb{P}[Right(\mathcal{A}) \to 1] - \mathbb{P}[Left(\mathcal{A}) \to 1].$$

$$InSec_Q^{PRP}(t, q) = \max_{\mathcal{A} \in A(t,q)} (Adv_Q^{PRP}(\mathcal{A})),$$

where $A(t, q)$ is a set of adversaries, whose running time does not exceed t and who uses no more than q oracle queries.

Remarks

- The quantity $InSec_Q^{PRP}(t, q)$ **directly depends** on the structure of the quasigroup Q.

Remarks

- The quantity $InSec_Q^{PRP}(t, q)$ **directly depends** on the structure of the quasigroup Q.
- We want it to be as small as possible for any given t, q.

Remarks

- The quantity $InSec_Q^{PRP}(t, q)$ **directly depends** on the structure of the quasigroup Q.
- We want it to be as small as possible for any given t, q.
- The inappropriate choice of quasigroup (i.e., $Q = \mathbb{Z}_N$) can make the problem trivial to solve.

Remarks

- The quantity $InSec_Q^{PRP}(t, q)$ **directly depends** on the structure of the quasigroup Q.
- We want it to be as small as possible for any given t, q.
- The inappropriate choice of quasigroup (i.e., $Q = \mathbb{Z}_N$) can make the problem trivial to solve.
- The problem might be hard for certain classes of quasigroups.

Remarks

- The quantity $InSec_Q^{PRP}(t, q)$ **directly depends** on the structure of the quasigroup Q.
- We want it to be as small as possible for any given t, q.
- The inappropriate choice of quasigroup (i.e., $Q = \mathbb{Z}_N$) can make the problem trivial to solve.
- The problem might be hard for certain classes of quasigroups.
- For instance, for polynomially complete quasigroups: the problem of deciding whether or not an equation over such a quasigroup has a solution is NP-complete.

Quasigroup based FPE

- Let Q be the quasigroup over the set $\mathtt{Dom} = \{0, 1\}^n$ for some «small» n.

Quasigroup based FPE

- Let Q be the quasigroup over the set $\mathbf{Dom} = \{0,1\}^n$ for some «small» n.
- Two-step procedure:
  1. given the key $k \in \mathbf{Keys}$ and the tweak $t \in \mathbf{Twk}$, use some keyed pseudorandom generator PRG to produce a sequence of «random-looking» and «independent» elements $q_i \in Q, i = 1, \dots, \lambda$, where λ is the parameter of the scheme and is chosen based on the quasigroup structure;

Quasigroup based FPE

- Let Q be the quasigroup over the set $\mathbf{Dom} = \{0,1\}^n$ for some «small» n.
- Two-step procedure:
  1. given the key $k \in \mathbf{Keys}$ and the tweak $t \in \mathbf{Twk}$, use some keyed pseudorandom generator PRG to produce a sequence of «random-looking» and «independent» elements $q_i \in Q, i = 1, \dots, \lambda$, where λ is the parameter of the scheme and is chosen based on the quasigroup structure;
  2. encrypt the message $m \in \mathbf{Dom}$ using quasigroup operation.

Possible operations

Some possible variants might be:

Possible operations

Some possible variants might be:

$$m \to L_{q_\lambda}(\dots L_{q_1}(m)\dots) = q_\lambda \circ (\dots \circ q_2 \circ (q_1 \circ m)\dots), \tag{1}$$

Possible operations

Some possible variants might be:

$$m \to L_{q_\lambda}(\dots L_{q_1}(m)\dots) = q_\lambda \circ (\dots \circ q_2 \circ (q_1 \circ m)\dots), \tag{1}$$

$$m \to R_{q_\lambda}(\dots R_{q_1}(m)\dots) = (\dots (m \circ q_1) \circ q_2 \dots) \circ q_\lambda, \tag{2}$$

Possible operations

Some possible variants might be:

$$m \to L_{q_\lambda}(\ldots L_{q_1}(m)\ldots) = q_\lambda \circ (\ldots \circ q_2 \circ (q_1 \circ m)\ldots), \tag{1}$$

$$m \to R_{q_\lambda}(\ldots R_{q_1}(m)\ldots) = (\ldots(m \circ q_1) \circ q_2 \ldots) \circ q_\lambda, \tag{2}$$

$$m \to D_{q_\lambda}(\ldots D_{q_1}(m)\ldots). \tag{3}$$

The operation D_{q_i} equals L_{q_i} if i-th bit of output of some random generator (for instance, based on values k and t) is equal to 0, and R_{q_i} otherwise.

Quasigroup based FPE: theorem

Theorem
Let q_t be the maximal number of different tweaks, q_e be the maximal number of encryption queries per tweak, t is the number of operations (running time). Then:

$$InSec^{TPRP}(t, q_t, q_e) \leq$$

$$\leq InSec^{PRG}(q_t, t + \lambda q_t q_e) + q_t InSec_Q^{PRP}(q_e, t + (1 + \lambda^2) q_e q_t).$$

Conclusions

Further areas of research may include:

Conclusions

Further areas of research may include:

1. Consideration of specific classes of quasigroups as a basis for proposed cryptosystem (with an emphasis on the polynomially complete quasigroups);

Conclusions

Further areas of research may include:

1. Consideration of specific classes of quasigroups as a basis for proposed cryptosystem (with an emphasis on the polynomially complete quasigroups);
2. Estimating the hardness of quasigroup problem based on existing results on NP-completeness;

Conclusions

Further areas of research may include:

1. Consideration of specific classes of quasigroups as a basis for proposed cryptosystem (with an emphasis on the polynomially complete quasigroups);
2. Estimating the hardness of quasigroup problem based on existing results on NP-completeness;
3. Implementing the cryptosystem over specific quasigroups and estimating statistical properties of resulting algorithms;

Summary

In this report:

1. FPE: statement of the problem;
2. Proposed solutions;
3. Attacks;
4. Quasigroup based FPE;