Cybersecurity

RESEARCH

Open Access

Generic attacks on small-state stream cipher constructions in the multi-user setting

Jianfu Huang[1], Ye Luo[1], Qinggan Fu[1], Yincen Chen[1], Chao Wang[1] and Ling Song[1,2*]

Abstract

Small-state stream ciphers (SSCs), which violate the principle that the state size should exceed the key size by a factor of two, still demonstrate robust security properties while maintaining a lightweight design. These ciphers can be classified into several constructions and their basic security requirement is to resist generic attacks, i.e., the time–memory–data tradeoff (TMDTO) attack. In this paper, we investigate the security of small-state constructions in the multi-user setting. Based on it, the TMDTO distinguishing attack and the TMDTO key recovery attack are developed for such a setting. It is shown that SSCs which continuously use the key can not resist the TMDTO distinguishing attack. Moreover, SSCs based on the continuous-IV-key-use construction cannot withstand the TMDTO key recovery attack when the key length is shorter than the IV length, no matter whether the keystream length is limited or not. Finally, we apply these two generic attacks to TinyJAMBU and DRACO in the multi-user setting. The TMDTO distinguishing attack on TinyJAMBU with a 128-bit key can be mounted with time, memory, and data complexities of 2^{64}, 2^{48}, and 2^{32}, respectively. This attack is comparable with a recent work on ToSC 2022, where partial key bits of TinyJAMBU are recovered with more than 2^{50} users (or keys). As DRACO's IV length is smaller than its key length, it is vulnerable to the TMDTO key recovery attack. The resulting attack has a time and memory complexity of both 2^{112}, which means DRACO does not provide 128-bit security in the multi-user setting.

Keywords Small-state stream ciphers, TMDTO attacks, Multi-user setting

Introduction

Stream ciphers are an important symmetric scheme. They use a key and an initial vector (IV) to generate a keystream for encryption and decryption. They are known for their high speed and low hardware complexity, making them important in digital communications, e.g., the E0 stream cipher for Bluetooth systems (Jiao et al. 2020).

Stream ciphers typically have two phases: initialization and keystream generation. In the first phase, the state is initialized with a key and an IV and then updated via an update function iteratively. In the second phase, the update function further obfuscates the state, and the output function generates the keystream. We call stream cryptographic ciphers self-synchronizing ciphers in which the generation of the keystream is influenced by the plaintext, and we call those synchronizing ciphers in which the keystream and the plaintext are independent (Rueppel 1986). Note that the key or the IV is usually not influencing the second phase.

A stream cipher of high quality should possess the ability to withstand all known forms of attacks, such as algebraic attack (Shannon 1949), guess-and-determine attack (Hawkes and Rose 2002), cube attack (Dinur and Shamir 2009), fault attack (Biham and Shamir 1997), time-memory tradeoff (TMTO) attack (Hellman 1980), and so on. Among these attacks, only the TMTO attack is generic for its high threat to various types of cryptographic

*Correspondence:
Ling Song
songling.qs@gmail.com
[1] College of Cyber Security, Jinan University, Guangzhou 510632, China
[2] National Joint Engineering Research Center of Network Security Detection and Protection Technology, Jinan University, Guangzhou 510632, China

Springer Open

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

algorithms without knowing their special structures and implementations.

The TMTO attack, proposed by Hellman (1980) in 1980, aims at recovering a key from plenty of plaintexts and ciphertexts. The attack balances the cost of time and memory through the tradeoff curve, which enables good performance in both time and memory complexity. Later, Baggage and Golic proposed a Time–memory–data tradeoff (TMDTO) attack (Babbage 1995; Golic 1997) which is a generalization of TMTO attack. It increases the amount of required data to get a better tradeoff. And in 2000, Biryukov and Shamir (Biryukov and Shamir 2000) improved the TMDTO attack for stream ciphers with different tradeoff curves and fewer disk operations. While the TMDTO attack is applied to a stream cipher, the objective is to retrieve the cipher's internal state in order to generate the subsequent keystream. In 2007, Håkan Englund et al. used a distinguishing attack to break Pomaranch stream cipher (Englund et al. 2007a, b). This attack can not only distinguish whether a sequence is random or ciphertext but also recover the remaining keystream sequence to get the whole plaintext in some special cases. In 2018, this attack was refined by Hamann et al. (2018).

Stream ciphers are composed of components like Boolean functions, S-boxes, Nonlinear-feedback shift register (NFSR), and so on. No matter how one carefully chooses these components for a stream cipher, it is hard to break the rule that the size of the state is twofold compared to that of the keystream, as derived in Babbage (1995). This becomes a hurdle for designing lightweight stream ciphers.

To break that limit, continuous-key-use (CKEY) construction has been proposed with Sprout (Armknecht and Mikhalev 2015). The CKEY construction continuously involves the key in the phase of the keystream generation, which can foil the TMDTO internal state attack. However, Sprout-like ciphers, such as Plantlet (Vasily et al. 2016) and Fruit (Amin and Honggang 2018), are vulnerable to the TMDTO distinguishing attack, as shown in Hamann et al. (2018). Later, to resist these distinguishing attacks, the continuous-IV-use (CIV) construction is proposed (Hamann et al. 2017), which continuously involves the IV instead in the phase of the keystream generation. The CIV construction together with limiting the length of the keystream can avoid the sliding property of the cipher and reduce the data that the above attacks need. Even though the CIV construction can resist the distinguishing attack, it is potentially vulnerable to other attacks as the IV is publicly known (Amin et al. 2019). Due to this, the CIV construction has not been used in real scenarios. Further, the construction continuing using both key and IV, which is called the CIVK construction, was proposed (Hamann et al. 2018). Even though the CIVK construction behaves heavily in many cryptosystem instances, it resists TMDTO attacks well and is considered as a new generic scheme. For example, the DRACO stream cipher (Hamann et al. 2022), which follows the CIVK construction, has been proposed in 2022. We call the stream cipher based on these constructions mentioned above the small-state stream cipher.

The growth of the Internet of Things (IoT) enables us to interact with a multitude of physical objects and exchange data through the Internet, enhancing our daily lives. To maintain secure communication in environments with limited resources, a variety of lightweight ciphers are employed in IoT-based applications (Philip and Vaithiyanathan 2017; Sehrawat and Gill 2018; Shah and Engineer 2019). Many types of IoT-based applications are facing explosive growth in users, and each of them is used with plenty of keys from users, such as Bluetooth, WiFi, RFID (Radio Frequency Identification), and so on Seliem et al. (2018). Naturally, the concept of multi-user security naturally arises. This notion for public-key encryption was put forth by Bellare et al. (2000), who observed that it could be exhibited in the multi-user setting. Afterward, an increasing number of works, such as Mouha and Luykx (2015), Tessaro (2015), Hoang and Tessaro (2016), Bellare and Tackmann (2016), Hoang and Tessaro (2017), and Bose et al. (2018), had concentrated on the symmetric cipher analyses on the multi-user security and the concept has become recognized as a more practical security goal. In 2016, Bellare and Tackmann (2016) researched the multi-user security of authenticated encryption. They considered the multi-user security of symmetric encryption and introduced two new concepts: indistinguishability security and key-recovery security. Recently, Muzhou et al. (2022) presented attacks under the multi-user setting which can bypass the TinyJAMBU cipher's restriction of the data per key. With this setting, a large number of attacks that are limited by the amount of available data will be revitalized. Thus, the small-state construction, which can thwart the TMDTO attack by restricting the keystream in the single-user setting, lacks the evidence to claim that the cryptosystem is secure in a realistic case where a multi-user setting is possible.

Our contribution

In this work, we investigate the feasibility of TMDTO attacks on small-state stream ciphers in the muti-user setting. By exploiting data obtained from different users, we present a generic TMDTO distinguishing attack on the CKEY construction and a TMDTO key recovery attack on the CIVK construction. Our paper reveals that in a multi-user setting, CKEY stream ciphers with

restricted keystream length fail to deliver adequate security. The constraint on the keystream length has an effect on both the memory complexity and the number of users necessary for an attack, while the time complexity of the attack remains unchanged. The TMDTO key recovery attack demonstrates that a stream cipher cannot remain secure through the restriction of its keystream length if its key length is shorter than its IV length. The complexity of this attack is associated with the ciphers' key length and IV length. The limit of keystream affects the number of users which our attacks need but the complexity.

Moreover, we apply these two attacks to TinyJAMBU and DRACO respectively, and find their vulnerability against TMDTO attacks in the multi-user setting. We bypass the limit of keystream length and successfully mount the TMDTO distinguishing attack on TinyJAMBU with time, memory, and data complexities of 2^{64}, 2^{48}, and 2^{32}, respectively. This attack is comparable with a recent work on ToSC 2022, where partial key bits of TinyJAMBU are recovered with more than 2^{50} users (or keys). The TMDTO key recovery attack on DRACO has a time and memory complexity of both 2^{112}, which means DRACO does not provide 128-bit security in the multi-user setting. Besides, we propose a new CIVK schema that is different from that of DRACO. With this construction, small-state ciphers have a better property of the TMDTO attacks resistance in the multi-user setting.

Outline

The structure of this paper is as follows for the remaining sections. In "Preliminary" section, we review TMDTO attacks, the multi-user setting, TMDTO distinguishing attacks, and TMDTO key recovery attacks. In "TMDTO attacks on small-state constructions in the multi-user setting" section, we propose TMDTO attacks in the multi-user setting, including the TMDTO distinguishing attack and the TMDTO key recovery attack. To demonstrate the applicability of the new attacks in the multi-user setting, we present a distinguishing attack on TinyJAMBU and a key recovery attack on DRACO in "Applications" section. Finally, "Conclusion" section is the conclusion of the paper.

Preliminary

Stream ciphers and small-state stream ciphers

Definition 1 (*Stream ciphers*) Stream ciphers are cryptosystems that use a time-dependent function to encrypt plaintexts. They comprise an internal state $S \in \{0,1\}^{\ell_s}$, an update function ϕ $\{0,1\}^{\ell_s} \rightarrow \{0,1\}^{\ell_s}$ and an output function α $\{0,1\}^{\ell_s} \rightarrow \{0,1\}^{\ell_o}$, where ℓ_s, ℓ_o are positive integers. The time-dependent function, which is made up of ϕ and α, takes a key K and an IV as inputs and updates S by the function ϕ (perhaps together with α) step by step, as illustrated in Fig. 1.

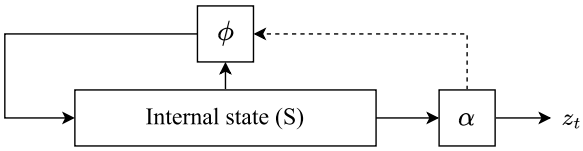

Fig. 1 The general construction of stream ciphers

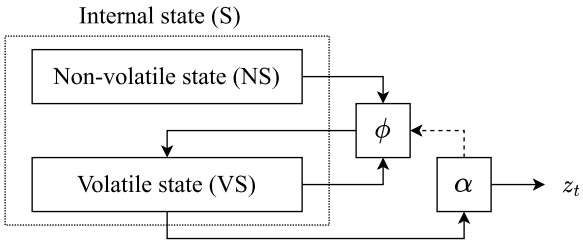

Fig. 2 The general construction of small-state stream ciphers

TMDTO attacks impose the constraint that the size of the state is twofold compared to that of the keystream. This inspires researchers to design special stream ciphers which can resist TMDTO attacks and maintain lightweight at the same time.

Small-state stream ciphers (SSCs), whose internal state contains an unstable state and a non-volatile instead of only an unstable state, use cheaper components such as ROM to construct the non-volatile state of the cipher while its volatile state is smaller than that of the general stream cipher. The general construction of SSCs is shown in Fig. 2 and examples of such stream ciphers include Sprout (Armknecht and Mikhalev 2015), Plantlet (Vasily et al. 2016) and Fruit (Amin and Honggang 2018).

Definition 2 (*Small-state stream ciphers*) Small-state stream ciphers (SSCs) are a type of keystream generator, which uses keystreams to encrypt plaintexts. It is made up of an internal state $S \in \{0,1\}^{\ell_s}$, an update function $\phi : \{0,1\}^{\ell_s} \rightarrow \{0,1\}^{\ell_{vs}}$ and an output function α $\{0,1\}^{\ell_{vs}} \rightarrow \{0,1\}^{\ell_o}$. S has two parts, i.e., a volatile state $VS \in \{0,1\}^{\ell_{vs}}$ and a non-volatile state $NS \in \{0,1\}^{\ell_{ns}}$. Note that ℓ_s, ℓ_{vs}, ℓ_{ns} and ℓ_o are positive integers. A key and an IV are accepted as inputs and the VS is updated by the function ϕ (perhaps together with α) during the encryption at each clock while the NS is fixed.

According to the treatment of the key and the IV, we can classify SSCs into the following categories (Hamann et al. 2019; Amin et al. 2019).

The continuous-Key-Use (CKEY) construction Loading the key into the *NS* during initialization and thus, the key gets involved in the update of the *VS*.

The continuous-IV-Use (CIV) construction Loading the IV into the *NS* during initialization and thus, the IV gets involved in the update of the *VS*.

The continuous-IV-Key-Use (CIVK) construction Loading both of the key and the IV into the *NS* during initialization and thus, the key and the IV get involved in the update of the *VS*.

The Lizard-like construction This small-state construction, as shown in Fig. 3, is different from the constructions described above, which only continuously use the key in a part of phases of the cryptosystem such as initialization.

Time–memory–data tradeoff attacks

We briefly revisit the relevant details of TMDTO attacks.

First of all, we introduce some notations.

- P: the time complexity of the pre-computation phase;
- T: the time complexity of the online phase;
- M: the memory complexity of the pre-computation phase;
- D: the data complexity of the online phase;
- N: the size of the state space;
- n: the length of the state which is equal to $\log_2 N$.
- $K \in \{0,1\}^{\ell_k}$: the ℓ_k-bit secret key.

In 1995, Babbage (1995) and Golić (1997) proposed a TMDTO attack called BG-TMDTO against stream ciphers. The attack has a pre-computation phase and an online phase. The attacker computes n-bit keystream prefixes corresponding to M initial states and stores them in a hash table on the procession of pre-computation. During the online phase, the attacker catches D keystream blocks and finds collisions by looking up the hash table. Actually, keystream blocks can be extracted from a long keystream with the method of sliding the window. If the collision happens, the inner state could be recovered with a high probability, with which one can predict the following keystream and recover the secret key as well. While $MD = N$, collisions could be found with high probability, resulting in a tradeoff curve of $TM = N$.

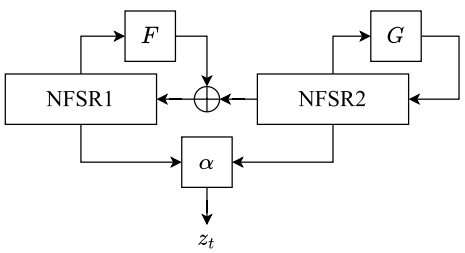

Fig. 3 Lizard-like small-state stream ciphers

We can get a special point $T = M = N^{\frac{1}{2}}$ from the tradeoff curve. When ℓ_k is less than half of n, the cipher's security is not guaranteed. Therefore, designers of a general stream cipher follow a guideline that the internal state length n of the cipher, defined in Definition 1, is at least twofold compared to the key length ℓ_k.

We can also expand this TMDTO attack to the TMDTO distinguishing attack and the TMDTO key recovery attack.

TMDTO distinguishing attacks

TMDTO distinguishing attacks were first proposed by Englund et al. (2007b) and was put forward in 2018 (Hamann et al. 2018), which enables these attacks to apply to more general stream ciphers by constructing attacks in different ways. In this attack scenario, the attacker tells whether a sequence is generated by a cryptosystem or random oracle. We call it `CIPHER` when we can identify from a large number of sequences that a keystream is generated by a particular cryptosystem and otherwise, we call it `RANDOM`.

This attack relies on the near-injectivity of IVs and the randomness of the initial state. It can be described in two steps.

Step 1 By sliding window, the attacker obtains $2^{\frac{n}{2}}$ keystream blocks and stores them in a table $\mathcal{H}$, which is generated with a fixed key and different IVs. If a collision occurs during this step, we can already distinguish `CIPHER` and stop.

Step 2 Capture D keystream prefixes generated by different IVs and look for a collision in $\mathcal{H}$ from Step 1. When a collision is found, we can identify it as `CIPHER` and the process can be halted.

To ensure that collisions in keystream blocks are due to colliding inner states, the size of the keystream blocks is made slightly larger than that of the internal state. Hence, we can further compare the subsequent keystream blocks corresponding to the two IVs, if they are the same, `CIPHER`. Otherwise, `RANDOM`.

Obviously, the memory complexity of this attack is $2^{\frac{n}{2}}$. A collision occurs in the table built in Step 1 when we intercept $D = 2^{\frac{n}{2}}$ keystream prefixes. Thus, the time complexity is also $2^{\frac{n}{2}}$. Note that we may fail to mount this attack on the cipher which limits keystream generated by per key or pair of (*key*, *IV*) because of the lacking of data, in other words, $D \leq 2^{\frac{n}{2}}$.

TMDTO key recovery attacks

TMDTO key recovery attacks focus on retrieving a secret key of a stream cipher. The process of these attacks is similar to BG-TMDTO (Hong and Sarkar 2005).

Step 1 (offline) Generate $2^{\frac{n}{2}}$ keystream prefixes corresponding to different keys and IVs. Save these keystream prefixes and the corresponding key and IV in a hash table $\mathcal{H}$. Note that the length of the keystream prefix is not less than the sum of the key size and the IV size.

Step 2 (online) Intercept D keystream prefixes and find collisions in $\mathcal{H}$. If a collision happens, we can recover the key and the IV corresponding with the keystream prefix.

As we describe above, assuming that we want to attack with high probability, D should not be less than $2^{\frac{n}{2}}$. The time complexity is equal to D and the memory complexity is $2^{\frac{n}{2}}$. And we should note that these attacks are thwarted when the length of the IV is not less than that of the key.

Small-state stream ciphers and TMDTO attacks

Small-state stream ciphers break the design guideline because of TMDTO attacks, which maintain lightweight and high security at the same time.

The CKEY stream cipher loads a key into the *NS* during initialization and thus, the key gets involved in updating the *VS* through the update function ϕ, which maps to shifted versions of the same keystream with different secret keys. In other words, the keystreams generated by different keys can result in the subsequent keystreams having the same bits due to collisions in the corresponding internal states at a certain bit. More specifically, the TMDTO attacks, which aim to recover the *VS* by finding collisions from keystream, are thwarted because the collision between two ℓ_{vs}-bit keystream blocks do not mean that the corresponding *VS*s collide. Additionally, the complexity of finding a collision in ℓ_s-bit keystream blocks to recover the S exceeds the security bound.

However, the CKEY stream cipher can not resist the TMDTO distinguishing attack as described in "TMDTO distinguishing attacks" section, since the shifted version of the same keystream with different IVs. The cipher based on the CIV or CIVK construction together with the limit of the keystream per key or pair of (*key*, *IV*) can actually thwart TMDTO attacks since the attacker can not obtain enough data to find collisions with low complexity.

The cipher based on the CIV construction loads an IV, instead of a key, into the *NS*. Thus, the IV is continuously used in the update of the *VS* through the function ϕ. In this way, the TMDTO distinguishing attack may fail because two keystream blocks generated with one key may map to different *VS*s generated with different IVs, even if these two keystream blocks are the same.

The ciphers based on the CIVK construction load both an IV and a key into the *NS* and then both of them get involved in the update of the *VS* by the function ϕ. This construction combines the advantages of the constructions mentioned above, which can resist the TMDTO internal state recovery attack as well.

TMDTO attacks on small-state constructions in the multi-user setting

The security in the multi-user setting

Before we describe our attack in the multi-user setting, we introduce some symbols as follows.

- The secret key $K \in \{0,1\}^{\ell_k}$
- The space of the internal state N
- The size of the internal state $n = \log_2 N$
- The hash table $\mathcal{H}$
- The initial vector $IV \in \{0,1\}^{\ell_{IV}}$
- Limit 2^{λ}-bit keystream per key or pair of (*key*, *IV*)

A stream cipher cryptosystem can be considered as an oracle $\mathcal{O}$ which consists of encryption and decryption, where the key space of $\mathcal{O}$ is $\{0,1\}^{\ell_k}$. When users A and B choose a key K, A can query $\mathcal{O}$ for encryption under key K to send a message $\mathcal{M}$ to B. And then B can query decryption under the same key K from $\mathcal{O}$ and get the message $\mathcal{M}$.

Definition 3 (*Single-user security*) An adversary $\mathcal{A}$ can request encryption from an oracle $\mathcal{O}$. The oracle $\mathcal{O}$ uses a key K that is random, independent, and unknown to $\mathcal{A}$. For messages $\mathcal{M}$ chosen by $\mathcal{A}$, the corresponding ciphertexts under K can be obtained from $\mathcal{O}$. The objective is to distinguish if the oracle $\mathcal{O}$ is `CIPHER` or `RANDOM`, and potentially recover K.

However, many systems and applications are multi-user oriented in the real world, such as broadcasting messages on the Internet, which leads to the concept of the multi-user setting.

Definition 4 (*Multi-user setting*) Each user A_i $(i > 1)$ holds a random, independent and different key K_i and a sender B send message m_i to A_i, encrypted under K_i through $\mathcal{O}$. This setting can also be called the multi-key setting while the users do not change their key during the communication.

Definition 5 (*Multi-user security*) An adversary $\mathcal{A}$ can query encryption from the oracle $\mathcal{O}$. The oracle $\mathcal{O}$ uses random and independent secret keys K_i $(i = 0, 1, \ldots)$ owned by different users. For messages $\mathcal{M}$ chosen by $\mathcal{A}$, $\mathcal{A}$ can get the corresponding ciphertexts under K_i from $\mathcal{O}$. The aim is to ascertain if the oracle $\mathcal{O}$ is CIPHER or RANDOM and even retrieve one of the secret keys.

A cryptosystem that is secure in the single-user setting may suffer threats in the multi-user setting. The security definitions for symmetric cryptosystem in the multi-user setting include indistinguishability security (Bellare and Tackmann 2016) and key recovery security (Bose et al. 2018).

Definition 6 (*Indistinguishability security*) The distinguishing game samples a random challenge, which is to distinguish whether an oracle $\mathcal{O}$ is CIPHER or RANDOM. The adversary $\mathcal{A}$ can utilize an oracle $\mathcal{N}$ to generate new user instances. Additionally, $\mathcal{A}$ has the ability to access an encryption oracle $\mathcal{O}$ with user i, an initial vector IV, and a fixed message M. We call the oracle $\mathcal{O}$ CIPHER when we can identify from a large number of sequences that a keystream is generated by a particular cryptosystem, and otherwise, we call it RANDOM.

Definition 7 (*Key recovery security*) The target of the attacker $\mathcal{A}$ is to output the key of any users. The same with Definition 6, $\mathcal{A}$ can access the oracle $\mathcal{N}$ and the encryption oracle $\mathcal{O}$. $\mathcal{A}$ queries the ciphertext C_i from $\mathcal{O}$ with a initial vector IV and a fixed message M under the user i. If a collision of C_i and a record computed offline occurs, we can recover the key of the user i.

The TMDTO distinguishing attack in the multi-user setting

As mentioned above, multi-user security is under the chosen plaintext attack. We can get sufficient data to carry out a distinguishing attack on SSCs in the multi-user setting.

Note that our attack is dedicated to the CKEY construction since the small-state construction which continuously uses the IV can foil the TMDTO distinguishing attack naturally.

Before we describe our distinguishing attack on the CKEY stream cipher, let Θ denote that we need 2^θ recipients in this scenario and let γ denote that we can store up to γ keystream blocks from each recipient in the hash table $\mathcal{H}$. Because of limiting the keystream length, we can not obtain enough data to mount the attack. Therefore, we need Θ recipients, each providing up to γ keystream blocks to build $\mathcal{H}$, to collide with a high possibility. Note that the size of the cipher's state is not exceeded by the size of the keystream block. We further describe Θ and γ in the following Table 1 corresponding to the two different ways of limiting the keystream length and two small-state constructions we attack.

Let us describe in detail how the distinguishing attack works, which is shown as Algorithm 2. $\mathcal{A}$ intercepts all keystream sequences by XORing the plaintexts and ciphertexts between both sides of communications. Applying Algorithm 1, $\mathcal{A}$ can get a large number of keystream blocks, which will be stored in $\mathcal{H}$ subsequently. As soon as one of them collides, stop it and we distinguish the system as CIPHER. If the space of $\mathcal{H}$ is full, $\mathcal{A}$ continuously intercepts keystreams, and at the same time, compares them with the data in $\mathcal{H}$. While the length of intercepted keystream in this step is equal to 2^λ bits and there is no collision found, we reset $\mathcal{H}$ and switch to the next key or pair of (*key*, *IV*) to repeat the above steps. When we change the key more than Θ times and no collision happens, we distinguish the oracle as RANDOM. Otherwise, we stop and distinguish it as CIPHER.

Algorithm 1 Sliding window

Input: Keystream KS;
The length of keystream KL;
The size of keystream block BL;
Output: A list of keystream blocks $\mathcal{L}$;
1: $\mathcal{L} \leftarrow [\,]$; // Declare a list $\mathcal{L}$.
2: **for** $i = 0$ *to* $KL - BL + 1$ **do**
3: $\quad \mathcal{L}[i] \leftarrow KS[i : i + BL - 1]$;
4: **end for**
5: **return** $\mathcal{L}$;

Algorithm 2 Distinguishing attack in the multi-user setting

Input: Hash table $\mathcal{H}$;
The max size of $\mathcal{H}$ γ;
The number of recipients Θ;
The max length of keystream per key or pair of (key, IV) 2^{λ};
The size of keystream block BL; The oracle $\mathcal{O}$;
Output: `CIPHER` or `RANDOM`;

$\mathcal{H} \leftarrow [\,]$; // Declare a hash table $\mathcal{H}$.
for each user **do**
  Intercept keystream between the user and $\mathcal{O}$;
  Get keystream blocks through the method of sliding window;
  for each keystream blocks **do**
    if this keystream blocks collides with an item in $\mathcal{H}$ **then**
      check their following keystream bits respectively;
      if their following keystream bits are the same as each other **then**
        return `CIPHER`;
      else
        return `RANDOM`;
      end if
    end if
    if the size of $\mathcal{H}$ is less than γ **then**
      Save this keystream block in $\mathcal{H}$;
    end if
  end for
  Empty $\mathcal{H}$;
end for
return `RANDOM`;

Let us analyze the feasibility of this attack. According to the birthday paradox, with around $2^{n/2}$ keystream blocks, each being n-bit long, the probability of a collision occurring is high. If θ is equal to 0, i.e., Θ is 1, then only one user is required to compromise the cryptosystem. Thus, we only talk about the other case.

Take the limit of keystream generated by each key as an example. The probability of finding collisions in $\mathcal{H}$ is not sufficient to mount the attack. Hence, we can sum all expectations of finding collisions in n-bit sequences to one to make our attack successful, i.e., we should search N collisions in n-bit sequences in which a collision occurs in high probability. While each user can generate up to 2^{λ}-bit keystream and search collisions about $2^{2\lambda}$ times, we can get a equation $2^n = 2^{2\lambda} \cdot 2^{\theta}$ for Θ users. Thus, we need $\Theta = 2^{n-2\lambda}$ users. The same is true of the limit of keystream bits per pair of (*key*, *IV*).

Table 1 The value of the two different ways on the number of recipients 2^{θ} and the size of the hash table γ

Various	Limit 2^{λ}-bit keystream per key	Limit 2^{λ}-bit keystream per pair of (*key*, *IV*)
θ	$max(0, (n - 2 \cdot \lambda))$	$max(0, (n - 2 \cdot \lambda \cdot \ell_{IV}))$
γ	$min(2^{\lambda}, 2^{n/2})$	$min(2^{\lambda} \cdot \ell_{IV}, 2^{n/2})$

Now we analyze the complexity of our attack. There are 2^{θ} users with 2^{γ} data respectively, which means about $2^{n/2}$ pairs keystream blocks are involved. Therefore, the time complexity is $2^{n/2}$. At the same time, we need to clear out all data of the table $\mathcal{H}$ to switch to the next user, so the memory complexity is 2^{γ}. Note that λ has an influence on the memory complexity and the number of users necessary for the attack, but not on the time complexity.

The TMDTO key recovery attack in the multi-user setting

Here we illustrate the attack. Suppose that we have a hash table $\mathcal{H}$ which stores blocks of (*key*, *IV* , *keystream prefix*). Note that the length of the keystream prefix is not less than the sum of ℓ_k and ℓ_{IV}. For convenience, we assume that the size of the keystream prefix is $\ell_k + \ell_{IV} + \epsilon$ where ϵ is a negligible number, and introduce our attack shown as follows.

Step 1 (offline) We generate the fragment at the beginning of keystream with different pairs of (*key*, *IV*), and put them in $\mathcal{H}$ until the size of $\mathcal{H}$ is $2^{\frac{\ell_k + \ell_{IV} + \epsilon}{2}}$. During this step, we should ensure that there is no collision in $\mathcal{H}$.

Step 2 (online) Intercept keystream, and at the same time, compare them with the data in the table. While

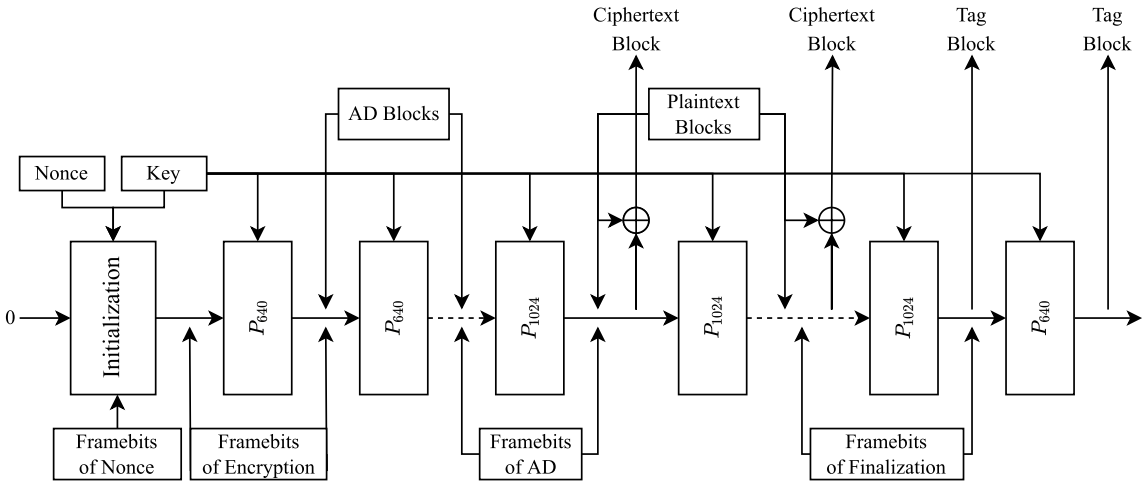

Fig. 4 The TinyJAMBU v2 mode for 128-bit state (Hongjun and Tao 2021). AD is denoted as association data. Denote P_i by processing the permutation i time(s)

a collision occurs, there is a high probability that we can recover the key.

As described above, our attack can break cryptosystems based on small-state constructions or plaintext dependant keystream generators. In addition, the limit of the keystream has no effect on our attack.

Now we analyze the feasibility of the attack, and we limit the keystream per pair of (*key*, *IV*) here. When $\ell_{IV} < \frac{\ell_k+\ell_{IV}}{2}$, in other words, the length of IV is shorter than that of the key, a single key can not generate more than $2^{\frac{\ell_k+\ell_{IV}+\epsilon}{2}}$ keystream prefixes, thus, we need $2^{\frac{\ell_k+\ell_{IV}+\epsilon}{2}-\ell_{IV}}$ chosen keys to provide a sufficient number of keystream prefixes to build $\mathcal{H}$. Hence, the success probability of this attack can be derived according to the birthday paradox. Both the time complexity and the memory complexity are $2^{\frac{\ell_k+\ell_{IV}+\epsilon}{2}}$. When $\ell_{IV} \geq \frac{\ell_k+\ell_{IV}}{2}$, in other words, the length of IV is not shorter than that of the key, our attack will not be feasible because the time complexity is higher than 2^{ℓ_k}. Note that 2^{λ} is generally not less than $\ell_k + \ell_{IV} + \epsilon$ in this scenario, by which the attack will not be influenced.

If we limit the keystream per key and $\ell_{IV} < \frac{\ell_k+\ell_{IV}}{2}$, we need to consider two possibilities. If $2^{\lambda} \geq (\ell_k + \ell_{IV} + \epsilon) \cdot 2^{\ell_{IV}}$, λ will not affect our attack which we can get enough data to mount the attack. Otherwise, a single key can only get up to $\frac{2^{\lambda}}{(\ell_k+\ell_{IV}+\epsilon)}$ keystream prefixes and we need more than $2^{\frac{\ell_k+\ell_{IV}+\epsilon}{2}} / \frac{2^{\lambda}}{(\ell_k+\ell_{IV}+\epsilon)}$ chosen keys to obtain enough data. However, λ only affects the number of chosen keys we need and will not affect the complexity of the attack. In this scenario, the time complexity is also $2^{\frac{\ell_k+\ell_{IV}+\epsilon}{2}}$ as well as the memory complexity.

Applications

The attack on TinyJAMBU

TinyJAMBU

TinyJAMBU (Hongjun and Tao 2019, 2021) is a lightweight scheme of authenticated encryption that uses a 128-bit keyed permutation. This permutation supports 3 types of key sizes which are 128 bits, 192 bits, and 256 bits. For our attack, we take TinyJAMBU v2 (Hongjun and Tao 2021) with a 128-bit key size and a 96-bit nonce size as an example to introduce it.

The authenticated encryption scheme has four parts: initialization, processing of associated data, encryption, and finalization, as depicted in Fig. 4.

Initialization This procession has two stages. Firstly, the key setup procession is to randomize the state by operating the keyed permutation. Secondly, the nonce setup procession has three steps, where 3 bits of the state are XORed with the framebits of a nonce, then the keyed permutation is processed, and XORing the state with 32-bit nonce blocks finally.

Processing the associated data Processing each 32-bit associated data block like the procession of nonce setup.

The encryption After processing the associated data, message M is divided into many 32-bit blocks to be encrypted. In each step, a 32-bit cipher block is generated by XORing a 32-bit message block and 32 bits of the state after updating the state by XORing the framebits of encryption and processing the keyed permutation. Meanwhile, the state is updated by XORed a message block.

The finalization A 64-bit authentication tag can be generated in two steps. Firstly, we generate the first 32-bit tag block by extracting 32-bit data from the state which has XORed the framebits of finalization and processed the

keyed permutation. And then, we generate the second 32-bit tag block in a similar way.

Note that TinyJAMBU stipulates that each secret key can process at most 2^{50} bytes of messages (*associated data*, *plaintext* / *ciphertext*), with each message being at least 8 bytes.

The TMDTO distinguishing attack on TinyJAMBU

As introduced above, when we use Hamann's distinguishing attack to break TinyJAMBU, it will not succeed because we do not have enough data. Specifically, the length of the keystream, which the attack needs, is $2^{n/2}$ bits, i.e., 2^{64} bits, but TinyJAMBU limits 2^{53} bits per key. Next, we will attack TinyJAMBU using the attack mentioned in "The TMDTO distinguishing attack in the multi-user setting" section.

To ensure the success of our attack, we make the assumption that all plaintext blocks corresponding to the keystream blocks generated with one key and many nonces are all zero or remain constant.

Note that if the attacker violates those assumptions, our attack can not process effectively because it is difficult to build the sorted keystream blocks table. In order to understand the assumptions and our attack on TinyJAMBU, we need to know some properties about TinyJAMBU as follows:

1. The plaintext block(32-bit) will not affect the state of the subsequent keyed permutation when a plaintext block is filled with zero or constant;
2. When the nonce is reused in TinyJAMBU mode, the role of the associated data is equivalent to that of the nonce.
3. When two keystream blocks generated by TinyJAMBU with the same parameters are the same, the following blocks are the same;
4. When a keystream block and a random sequence block are the same, the following blocks are the same with probability close to 2^{-32}.

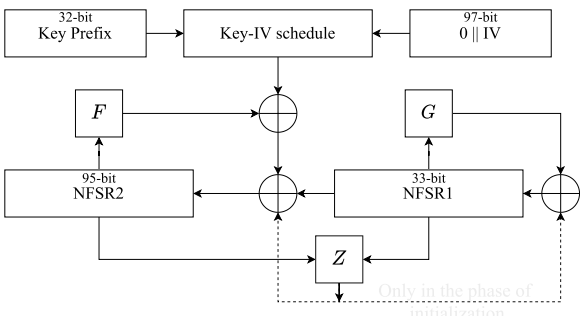

Fig. 5 The block diagram of DRACO

We should note that

1. The keystream length we intercept is 160 bits, i.e., five keystream blocks, which we can find collision through 128 bits and use 32 bits to verify.
2. TinyJAMBU limits 2^{53}-bit keystream per key. We can generate 2^{48} 32-bit keystream blocks, i.e., λ is 48, and get $\theta = max(0, n - 2 \cdot \lambda) = 32$ and $\gamma = min(2^{\lambda}, 2^{n/2}) = 2^{48}$.
3. Let $\mathcal{H}$ denote the hash table which can save at most γ keystream blocks generated by one key. It is obviously that λ is less than $\frac{n}{2}$. Thus, we must catch at least Θ user from user communications so that we can mount our attack successfully.

In this attack scenario, the probability of success can be derived from "The TMDTO key recovery attack in the multi-user setting" section. The complexities of time and memory are 2^{64} and 2^{48}, respectively. Meanwhile, the number of users that the attack needs is 2^{32}.

Recently, Muzhou et al. (2022) proposed a new statistical model on the extension of linear cryptanalysis to linear hulls and obtain cryptanalysis results on TinyJAMBU (Hongjun and Tao 2019, 2021) in ToSC 2022, where partial key bits are recovered. In that paper, the adversary needs to choose related keys between users, which is much stronger than the normal multi-user setting in our attacks. In addition, Muzhou et al. (2022) mounts partial key recovery attacks using more than 2^{50} users (or keys) while our work is a distinguishing attack but with only 2^{32} users (or keys) and also much lower time complexity.

The attack on DRACO

DRACO

The DRACO stream cipher (Hamann et al. 2022), proposed in IACR ToSC 2022, uses a 128-bit key and a 96-bit IV. It has a 128-bit inner state that is composed of 2 NFSRs with lengths of 95 bits and 33 bits, respectively.

As the figure is shown in Fig. 5, the *NS* consists of part of the key and full bits of the IV, and the volatile state is the 128-bit inner state. The feedback functions of NFSR1 and NFSR2 are denoted as *G* and *F* respectively. The output function is denoted as *Z*.

The process of generating a keystream is divided into two parts, state initialization, and keystream generation, where state initialization is also divided into two phases, key-IV loading, and Mixing. During the state initialization, we load a key prefix and an IV into the *NS* and load full bits of the key into the volatile state. Then, we process the mixing phase. In this phase, clocking the cipher 512 times without generating any bits, and updating the inner state by XORing the output of key-IV schedule, *F*, *G*, *Z*,

and the state of NFSR1 and NFSR2. Afterward, Z starts generating the keystream by processing the update function without incorporating Z's output. Note that DRACO limits 2^{32}-bit keystream per pair of (key, IV).

The TMDTO key recovery attack on DRACO

With the illustration of DRACO and our attack in "The TMDTO key recovery attack in the multi-user setting" section, the IV in DRACO is 96-bit which is less than the length of the key, which can pose a threat to DRACO.

To make our attack effective, we should generate 224-bit keystream prefixes per (key, IV) pair and save blocks, which consists of key, IV, and keystream prefix, into the hash table $\mathcal{H}$ offline. More specifically, as we can generate about 2^{46} keystream prefixes per pair of (key, IV), we should generate enough keystream prefixes with 2^{66} keys and save at least 2^{112} blocks into $\mathcal{H}$. According to "The TMDTO key recovery attack in the multi-user setting" section, we can compare the keystream prefixes that we catch online with the keystream prefixes saved in $\mathcal{H}$. If a collision occurs, we can recover the key.

The time complexity is 2^{112} which is the same as the memory complexity and less than 2^{128}, so our attack is valid. In addition, the number of the chosen keys, which the attack needs, is 2^{32}.

Note that, Subhadeep Banik et al. posed two TMDTO attacks (Banik 2022) on DRACO recently, which exploits the fact that The state update function utilizes only a minor fraction of the NS. However, our attack is not aimed at exploiting the vulnerability of the components but the CIVK construction under the security parameters of DRACO. Our attack shows that DRACO is not a good instance of the CIVK construction in resisting attacks under the multi-user setting.

CIVK schemes and TMDTO attacks

DRACO has shown that its CIVK scheme has excellent resistance to TMDTO attacks in single-user mode. While the IV size is less than the key size, DRACO is lack resistance to TMDTO attacks in the multi-user scenario. Therefore, we make improvements to the CIVK scheme based on DRACO and analyze it for TMDTO attacks security.

The IV size ℓ_{IV}-bit of our CIVK scheme is 128-bit which is the same as the key size ℓ_k-bit. NS, whose size is 128-bit, consists of a key prefix and an IV prefix whose lengths are ℓ_k^{pre}-bit and ℓ_{IV}^{pre}-bit respectively. Once the remaining key bits and full of the IV bits have been loaded into the internal state, our scheme proceeds through phases similar to those of DRACO. As for the limit of keystream length, our scheme allows up to $2^{64}-1$ bits to be generated per key, rather than per (key, IV) pair.

Conclusion

Our work reveals the fact that some of the small-state ciphers, which can resist TMDTO attacks excellently, are still vulnerable to TMDTO attacks in the multi-user scenario. We describe a distinguishing attack and a key recovery attack under the multi-user setting with several examples based on TMDTO attacks. The multi-user setting is reflecting the real usage of lightweight applications, such as the widespread use of RFID. Thus, in the multi-user setting, attackers have sufficient data available to mount TMDTO attacks. The stream cipher based on the CKEY construction is vulnerable to the TMDTO distinguishing attack in the multi-user scenario, even though it limits the keystream per key or pair of (key, IV). While the CIVK scheme can thwart not only the TMDTO inner state recovery attack but also the TMDTO distinguishing attack. Therefore, the cipher based on the CIVK construction should uphold the principle that the IV length cannot be less than the key length. For the designer, the CIVK construction, with the restriction of the keystream length, should be considered as a generic scheme to design a small-state cipher that can secure the stream cipher against TMDTO attacks in the real world.

Authors' contributions
All the authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China [grant number 62022036, 62132008, 62372213].

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 16 May 2023 Accepted: 30 August 2023
Published online: 08 October 2023

References

Amin GV, Honggang H (2018) Fruit-80: a secure ultra-lightweight stream cipher for constrained environments. Entropy 20(3):180. https://doi.org/10.3390/e20030180

Amin GV, Honggang H, Fujiang L (2019) On designing secure small-state stream ciphers against time-memory-data tradeoff attacks. Cryptology ePrint Archive, Preprint https://eprint.iacr.org/2019/670

Armknecht F, Mikhalev V (2015) On lightweight stream ciphers with shorter internal states. In: Fast software encryption—22nd international workshop—FSE 2015—Istanbul, Revised Selected Papers. Lecture Notes in Computer Science, vol 9054, pp 451–470. https://doi.org/10.1007/978-3-662-48116-5_22

Babbage SH (1995) Improved "exhaustive search" attacks on stream ciphers. In: European convention on security and detection 1995, pp 161–166. https://doi.org/10.1049/cp:19950490
Banik S (2022) Cryptanalysis of draco. IACR Transactions on symmetric cryptology, pp 92–104. https://doi.org/10.46586/tosc.v2022.i4.92-104
Bellare M, Tackmann B (2016) The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Robshaw M, Katz J (eds) Advances in cryptology—CRYPTO 2016—36th annual international cryptology conference, Santa Barbara, Part I. Lecture Notes in Computer Science, vol 9814, pp 247–276. https://doi.org/10.1007/978-3-662-53018-4_10
Bellare M, Boldyreva A, Micali S (2000) Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel B (ed) Advances in Cryptology—EUROCRYPT 2000, International conference on the theory and application of cryptographic techniques, Bruges, Lecture Notes in Computer Science, vol 1807, pp 259–274. doi: https://doi.org/10.1007/3-540-45539-6_18
Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: Jr. BSK (ed) Advances in cryptology—CRYPTO '97, 17th annual international cryptology conference, Santa Barbara, Lecture Notes in Computer Science, vol 1294, pp 513–525. doi: https://doi.org/10.1007/BFb0052259
Biryukov A, Shamir A (2000) Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Tatsuaki O (ed) Advances in cryptology—ASIACRYPT 2000. ASIACRYPT 2000. Lecture Notes in Computer Science, pp 1–13. doi: https://doi.org/10.1007/3-540-44448-3_1
Bose P, Hoang VT, Tessaro S (2018) Revisiting AES-GCM-SIV: multi-user security, faster key derivation, and better bounds. In: Nielsen JB, Rijmen V (eds) Advances in cryptology—EUROCRYPT 2018—37th annual international conference on the theory and applications of cryptographic techniques, Tel Aviv, Part I, pp 468–499. doi: https://doi.org/10.1007/978-3-319-78381-9_18
Dinur I, Shamir A (2009) Cube attacks on tweakable black box polynomials. In: Joux A (ed) Advances in Cryptology—EUROCRYPT 2009, 28th annual international conference on the theory and applications of cryptographic techniques, Cologne, Germany, Lecture Notes in Computer Science, vol 5479, pp 278–299. doi: https://doi.org/10.1007/978-3-642-01001-9_16
Englund H, Hell M, Johansson T (2007) A note on distinguishing attacks. In: Proceedings of the IEEE Information theory workshop on information theory for wireless networks, Solstrand, pp 1–4. doi: https://doi.org/10.1109/ITWITWN.2007.4318038
Englund H, Hell M, Johansson T (2007) Two general attacks on pomaranch-like keystream generators. In: Fast software encryption, 14th international workshop—FSE 2007—Luxembourg, Revised Selected Papers, pp 274–289. doi: https://doi.org/10.1007/978-3-540-74619-5_18
Golic JD (1997) Cryptanalysis of alleged A5 stream cipher. In: Fumy W (ed) Advances in cryptology—EUROCRYPT '97, International conference on the theory and application of cryptographic techniques, Konstanz, Lecture Notes in Computer Science, vol 1233, pp 239–255. doi: https://doi.org/10.1007/3-540-69053-0_17
Hamann M, Krause M, Meier W (2017) A note on stream ciphers that continuously use the IV. Cryptology ePrint Archive, Preprint https://eprint.iacr.org/2017/1172
Hamann M, Krause M, Meier W, Zhang B (2018) Design and analysis of small-state grain-like stream ciphers. Cryptogr Commun 10:803–834. https://doi.org/10.1007/s12095-017-0261-6
Hamann M, Krause M, Moch A (2020) Tight security bounds for generic stream cipher constructions. In: Paterson KG, Stebila D (eds) Selected Areas in cryptography—SAC 2019—26th international conference, Waterloo, pp 335–364. doi: https://doi.org/10.1007/978-3-030-38471-5_14
Hamann M, Moch A, Krause M, Mikhalev V (2022) The DRACO stream cipher: a power-efficient small-state stream cipher with full provable security against TMDTO attacks. IACR transactions on symmetric cryptology, pp 1–42. doi: https://doi.org/10.46586/tosc.v2022.i2.1-42
Hawkes P, Rose GG (2002) Guess-and-determine attacks on snow. In: Nyberg K, Heys HM (eds) Selected areas in cryptography, 9th annual international workshop, SAC 2002, St. John's, Newfoundland, Canada, Revised Papers. Lecture Notes in Computer Science, vol 2595, pp 37–46. doi: https://doi.org/10.1007/3-540-36492-7_4
Hellman M (1980) A cryptanalytic time-memory trade-off. IEEE Trans Inf Theory 26(4):401–406. https://doi.org/10.1109/TIT.1980.1056220
Hoang VT, Tessaro S (2016) Key-alternating ciphers and key-length extension: exact bounds and multi-user security. In: Robshaw M, Katz J (eds) Advances in cryptology—CRYPTO 2016—36th annual international cryptology conference, Santa Barbara, Part I. Lecture Notes in Computer Science, vol 9814, pp 3–32. doi: https://doi.org/10.1007/978-3-662-53018-4_1
Hoang VT, Tessaro S (2017) The multi-user security of double encryption. In: Coron J, Nielsen JB (eds) Advances in Cryptology—EUROCRYPT 2017—36th annual international conference on the theory and applications of cryptographic techniques, Paris, Part II. Lecture Notes in Computer Science, vol 10211, pp 381–411. doi: https://doi.org/10.1007/978-3-319-56614-6_13
Hong J, Sarkar P (2005) New applications of time memory data tradeoffs. In: Roy BK (ed) Advances in Cryptology—ASIACRYPT 2005, 11th international conference on the theory and application of cryptology and information security, Chennai, Lecture Notes in Computer Science, vol 3788, pp 353–372. doi: https://doi.org/10.1007/11593447_19
Hongjun W, Tao H (2019) TinyJAMBU: a family of lightweight authenticated encryption algorithms. The NIST lightweight cryptography competition. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/ip-statements/TinyJAMBU-Statements.pdf
Hongjun W, Tao H (2021) TinyJAMBU: A family of lightweight authenticated encryption algorithms (Version 2). The NIST lightweight cryptography competition. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/tinyjambu-spec-final.pdf
Jiao L, Hao Y, Feng D (2020) Stream cipher designs: a review. Sci China Inf Sci 63(3):1–25. https://doi.org/10.1007/s11432-018-9929-x
Mouha N, Luykx A (2015) Multi-key security: the even-mansour construction revisited. In: Gennaro R, Robshaw M (eds) Advances in Cryptology—CRYPTO 2015—35th annual cryptology conference, Santa Barbara, Part I. Lecture Notes in Computer Science, vol 9215, pp 209–223. doi: https://doi.org/10.1007/978-3-662-47989-6_10
Muzhou L, Nicky M, Ling S, Meiqin W (2022) Revisiting the extension of Matsui's algorithm 1 to linear hulls: application to TinyJAMBU. IACR transactions on symmetric cryptology, pp 161–200. doi: https://doi.org/10.46586/tosc.v2022.i2.161-200
Philip MA, Vaithiyanathan (2017) A survey on lightweight ciphers for IoT devices. In: 2017 international conference on technological advancements in power and energy (TAP Energy), pp 1–4. doi: https://doi.org/10.1109/TAPENERGY.2017.8397271
Rueppel RA (1986) Analysis and design of stream ciphers. Springer, Berlin, Heidelberg, pp 5–16. https://doi.org/10.1007/978-3-642-82865-2_2
Sehrawat D, Gill NS (2018) Lightweight block ciphers for IoT based applications: a review. Int J Appl Eng Res 13(5):2258–2270
Seliem M, Elgazzar K, Khalil K (2018) Towards privacy preserving IoT environments: a survey. Wirel Commun Mob Comput 2018:1032761. https://doi.org/10.1155/2018/1032761
Shah A, Engineer M (2019) A survey of lightweight cryptographic algorithms for IoT-based applications. In: Smart innovations in communication and computational sciences: proceedings of ICSICCS-2018, Springer, pp 283–293. doi: https://doi.org/10.1007/978-981-13-2414-7_27
Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28(4):656–715
Tessaro S (2015) Optimally secure block ciphers from ideal primitives. In: Iwata T, Cheon JH (eds) Advances in cryptology— ASIACRYPT 2015—21st international conference on the theory and application of cryptology and information security, Auckland, Part II. Lecture Notes in Computer Science, vol 9453, pp 437–462. doi: https://doi.org/10.1007/978-3-662-48800-3_18
Vasily M, Frederik A, Christian M (2016) On ciphers that continuously access the non-volatile key. IACR transactions on symmetric cryptology, pp 52–79. doi: https://doi.org/10.13154/tosc.v2016.i2.52-79

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.