Exact Security Analysis of Ascon

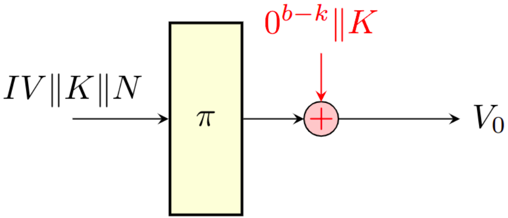
Bishwajit Chakraborty[1,2] Chandranan Dhar[1] Mridul Nandi[1]

[1]Indian Statistical Institute, Kolkata, India
[2]Nanyang Technological University, Singapore

Asiacrypt 2023
December 05, 2023

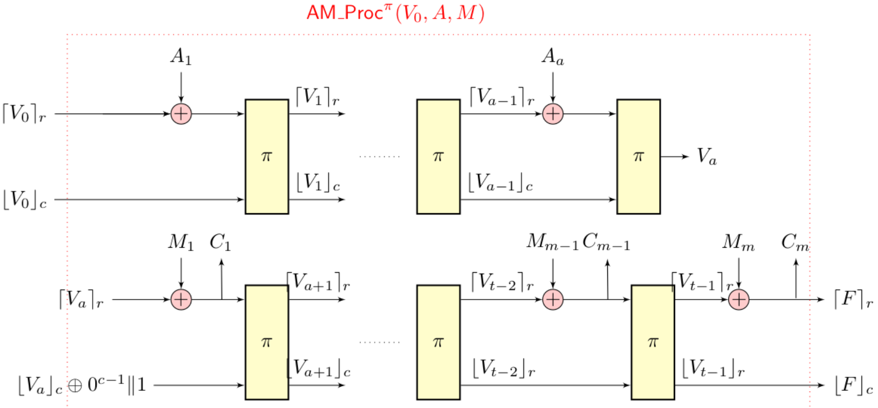
Presentation Overview

Divided into three steps. First step: Initialization.

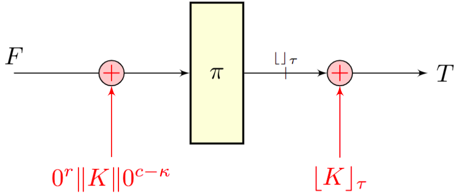


Difference with Generic Duplex: Key XOR-ed to the output.

Second Step: Processing Associated Data / Message / Ciphertext.



Third Step: Tag Generation / Verification.



Difference with Generic Duplex: Key XOR-ed to both input and output.

Existing Security Analyses

- Ascon lacks a dedicated security analysis. Existing studies consider it as a derivative of the Duplex construction.
- The best known bounds are of the order

$$DT/2^c$$

where D and T are the data and time complexities respectively, and c denotes the capacity of the underlying sponge.

Existing Security Analyses

- Ascon lacks a dedicated security analysis. Existing studies consider it as a derivative of the Duplex construction.
- The best known bounds are of the order

$$DT/2^c$$

where D and T are the data and time complexities respectively, and c denotes the capacity of the underlying sponge.

Security Model

- We analyze the security of Ascon in the random permutation model.
- The adversary can perform encryption, decryption, and (bi-directional) permutation queries in any order.
- We consider only nonce-respecting adversaries.
- We assume that the key size is at least as large as the tag size. This ensures the masking of the entire tag.

Security Model

- We analyze the security of Ascon in the random permutation model.
- The adversary can perform encryption, decryption, and (bi-directional) permutation queries in any order.
- We consider only nonce-respecting adversaries.
- We assume that the key size is at least as large as the tag size. This ensures the masking of the entire tag.

Security Model

- We analyze the security of Ascon in the random permutation model.
- The adversary can perform encryption, decryption, and (bi-directional) permutation queries in any order.
- We consider only nonce-respecting adversaries.
- We assume that the key size is at least as large as the tag size. This ensures the masking of the entire tag.

Security Model

- We analyze the security of Ascon in the random permutation model.
- The adversary can perform encryption, decryption, and (bi-directional) permutation queries in any order.
- We consider only nonce-respecting adversaries.
- We assume that the key size is at least as large as the tag size. This ensures the masking of the entire tag.

Main Theorem

Theorem

Let κ and τ denote the key-size and tag-size respectively. Then, for any adversary $\mathcal{A}$ with data complexity D and time complexity T, we have

$$\mathbf{Adv}^{\mathsf{AEAD}}_{\mathsf{Ascon}}(\mathcal{A}) = \mathcal{O}\left(\frac{T}{2^c} + \frac{T}{2^\kappa} + \frac{D}{2^\tau}\right).$$

Interpretation of the Result

with NIST parameters

NIST requirements: $D = 2^{53}, T = 2^{112}, \kappa \geq 2^{128}, \tau \geq 2^{64}$.

In light of NIST requirements, our bound shows that Ascon is secure even when capacity is reduced to 136 bits, and tag to just 64 bits.

This implies the Ascon AEAD can be made almost 50% more efficient without degrading its security.

Interpretation of the Result

with NIST parameters

NIST requirements: $D = 2^{53}, T = 2^{112}, \kappa \geq 2^{128}, \tau \geq 2^{64}$.

In light of NIST requirements, our bound shows that Ascon is secure even when capacity is reduced to 136 bits, and tag to just 64 bits.

This implies the Ascon AEAD can be made almost 50% more efficient without degrading its security.

Interpretation of the Result

with NIST parameters

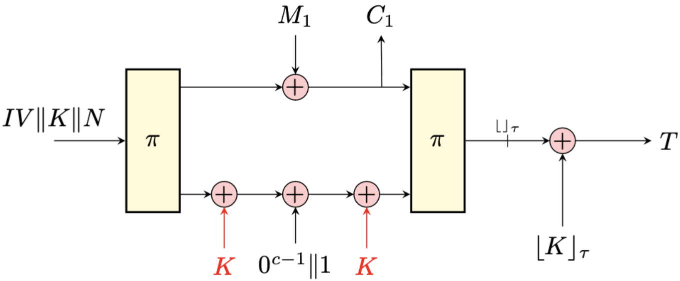
NIST requirements: $D = 2^{53}, T = 2^{112}, \kappa \geq 2^{128}, \tau \geq 2^{64}$.

In light of NIST requirements, our bound shows that Ascon is secure even when capacity is reduced to 136 bits, and tag to just 64 bits.

This implies the Ascon AEAD can be made almost 50% more efficient without degrading its security.

A Small Correction

In our original paper, it was mentioned that the capacity can be reduced to 128 bits. However, the analysis has a slight oversight.



If $c = \kappa$, then in the special case when $AD = \emptyset$, $|M| < r$, the keys in red cancel out each other. So, we revise with $c = 136$.

Can be found in Eprint archive 2023/775

Tightness of our Bounds

The bound that we achieve is tight:

- Attacks of the order $T/2^c$ can be constructed by observing state collisions in permutation queries.
- Generic key-guessing attacks in permutation queries are of the order $T/2^\kappa$.
- Generic tag-guessing attacks in online queries (mainly decryption queries) are of order $D/2^\tau$.

Tightness of our Bounds

The bound that we achieve is tight:

- Attacks of the order $T/2^c$ can be constructed by observing state collisions in permutation queries.
- Generic key-guessing attacks in permutation queries are of the order $T/2^\kappa$.
- Generic tag-guessing attacks in online queries (mainly decryption queries) are of order $D/2^\tau$.

Tightness of our Bounds

The bound that we achieve is tight:

- Attacks of the order $T/2^c$ can be constructed by observing state collisions in permutation queries.
- Generic key-guessing attacks in permutation queries are of the order $T/2^\kappa$.
- Generic tag-guessing attacks in online queries (mainly decryption queries) are of order $D/2^\tau$.

Proof Overview I

The Real World

We employ the H-coefficient technique for our proof.

In the real world, a key K and a random permutation Π are sampled independently. All queries are then responded to honestly.

Extended transcript consists of:

- all inputs and outputs corresponding to encryption, decryption, and primitive queries,
- all inputs and outputs of the permutation calls corresponding to encryption and decryption queries.

Proof Overview I

The Real World

We employ the H-coefficient technique for our proof.

In the real world, a key K and a random permutation Π are sampled independently. All queries are then responded to honestly.

Extended transcript consists of:

- all inputs and outputs corresponding to encryption, decryption, and primitive queries,
- all inputs and outputs of the permutation calls corresponding to encryption and decryption queries.

Proof Overview I

The Real World

We employ the H-coefficient technique for our proof.

In the real world, a key K and a random permutation Π are sampled independently. All queries are then responded to honestly.

Extended transcript consists of:

- all inputs and outputs corresponding to encryption, decryption, and primitive queries,
- all inputs and outputs of the permutation calls corresponding to encryption and decryption queries.

Proof Overview II
The Ideal World

The ideal world consists of online phase and offline phase.

Online phase:

- encryption queries responded to randomly,
- all decryption queries are rejected, and
- permutation queries are responded to faithfully.

Offline phase samples intermediate variables, and generates extended transcript.

Proof Overview II

The Ideal World

The ideal world consists of online phase and offline phase.

Online phase:

- encryption queries responded to randomly,
- all decryption queries are rejected, and
- permutation queries are responded to faithfully.

Offline phase samples intermediate variables, and generates extended transcript.

Proof Overview II

The Ideal World

The ideal world consists of online phase and offline phase.

Online phase:

- encryption queries responded to randomly,
- all decryption queries are rejected, and
- permutation queries are responded to faithfully.

Offline phase samples intermediate variables, and generates extended transcript.

Proceeds in stages:

1. Start with permutation query transcript P.
2. Sample intermediate variables for encryption queries to obtain permutation input-output pairs P_E.
3. Randomly extend P to P_1 by setting input-outputs for decryption queries.
   Set $P_2 := P_1 \cup P_E$.
4. Finally, sample key K. Set input-output pairs for initialization first and then the finalization phase.
   Update P_2 twice to obtain P_{fin}.

Proof Overview III

Offline Phase of the Ideal World

Proceeds in stages:

1. Start with permutation query transcript P.
2. Sample intermediate variables for encryption queries to obtain permutation input-output pairs P_E.
3. Randomly extend P to P_1 by setting input-outputs for decryption queries.
   Set $P_2 := P_1 \cup P_E$.
4. Finally, sample key K. Set input-output pairs for initialization first and then the finalization phase.
   Update P_2 twice to obtain P_{fin}.

Proof Overview III

Offline Phase of the Ideal World

Proceeds in stages:

1. Start with permutation query transcript P.
2. Sample intermediate variables for encryption queries to obtain permutation input-output pairs P_E.
3. Randomly extend P to P_1 by setting input-outputs for decryption queries.
   Set $P_2 := P_1 \cup P_E$.
4. Finally, sample key K. Set input-output pairs for initialization first and then the finalization phase.
   Update P_2 twice to obtain P_{fin}.

Proof Overview III

Offline Phase of the Ideal World

Proceeds in stages:

1. Start with permutation query transcript P.
2. Sample intermediate variables for encryption queries to obtain permutation input-output pairs P_E.
3. Randomly extend P to P_1 by setting input-outputs for decryption queries.
   Set $P_2 := P_1 \cup P_E$.
4. Finally, sample key K. Set input-output pairs for initialization first and then the finalization phase.
   Update P_2 twice to obtain P_{fin}.

Proof Overview IV

Bad Events

In the offline phase of the ideal world, bad events occur when

- Variables sampled are not permutation-compatible.
  Order of event: $T/2^c$ and $T/2^\kappa$.
- We have a correct forging.
  Order of event: Not significant.
- Decryption queries are not rejected.
  Order of event: $D/2^\tau$.

Proof Overview IV

Bad Events

In the offline phase of the ideal world, bad events occur when

- Variables sampled are not permutation-compatible.
  Order of event: $T/2^c$ and $T/2^\kappa$.
- We have a correct forging.
  Order of event: Not significant.
- Decryption queries are not rejected.
  Order of event: $D/2^\tau$.

Proof Overview IV

Bad Events

In the offline phase of the ideal world, bad events occur when

- Variables sampled are not permutation-compatible.
  Order of event: $T/2^c$ and $T/2^\kappa$.
- We have a correct forging.
  Order of event: Not significant.
- Decryption queries are not rejected.
  Order of event: $D/2^\tau$.

Conclusion
and possible extensions

- Key enabler of proof: double-keyed finalization of Ascon.
- Analysis does not directly apply to other keyed Sponge-based constructions, even with weaker security. Best-known bound for generic constructions still $DT/2^c$, which is not tight.
- In multi-user setting, we think the bound degrades to $\mu T/2^\kappa$, where μ denotes the number of users. Separate analysis required. Also interesting would be having a tight bound for nonce-misuse authenticity. Currently working on them.

Conclusion

and possible extensions

- Key enabler of proof: double-keyed finalization of Ascon.
- Analysis does not directly apply to other keyed Sponge-based constructions, even with weaker security. Best-known bound for generic constructions still $DT/2^c$, which is not tight.
- In multi-user setting, we think the bound degrades to $\mu T/2^\kappa$, where μ denotes the number of users. Separate analysis required. Also interesting would be having a tight bound for nonce-misuse authenticity. Currently working on them.

Conclusion
and possible extensions

- Key enabler of proof: double-keyed finalization of Ascon.
- Analysis does not directly apply to other keyed Sponge-based constructions, even with weaker security. Best-known bound for generic constructions still $DT/2^c$, which is not tight.
- In multi-user setting, we think the bound degrades to $\mu T/2^\kappa$, where μ denotes the number of users. Separate analysis required. Also interesting would be having a tight bound for nonce-misuse authenticity. Currently working on them.

Thank You!

Questions? Comments?