

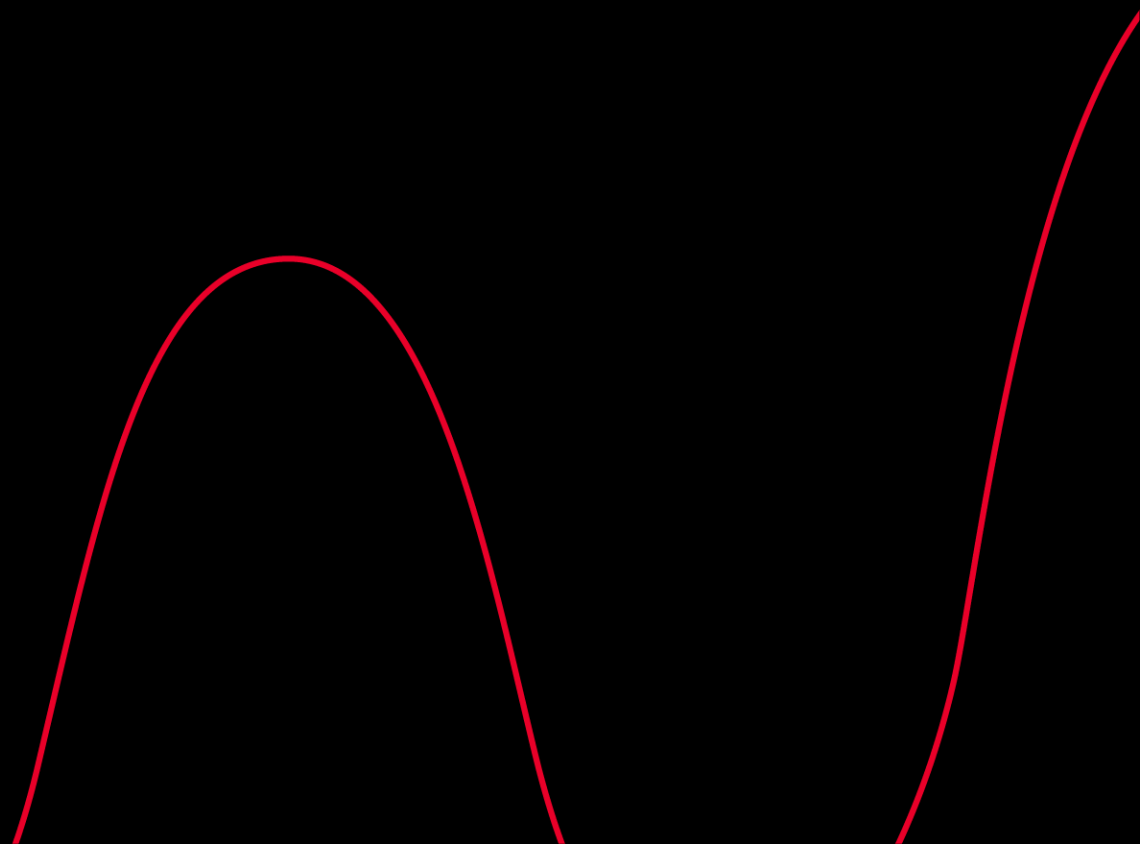
Weird crypto protocols I have seen in real-world products

Turid Herland, Senior Security Analyst
Norsk kryptoseminar 2025

Definition:

A **weird crypto protocol** is a crypto protocol that I had never heard about before I came across it in a project

Butterfly Key Expansion



Butterfly Key Expansion

What?

- A method for deriving several ECC keypairs from one seed keypair.

How?

- Given:
 - Seed keypair $\{a, A = aG\}$
 - G is a base point of order l
- New keypairs:
 - $\{b_i = a + x_i, B_i = A + x_i \cdot G\}$
 - The x_i are random integers mod l

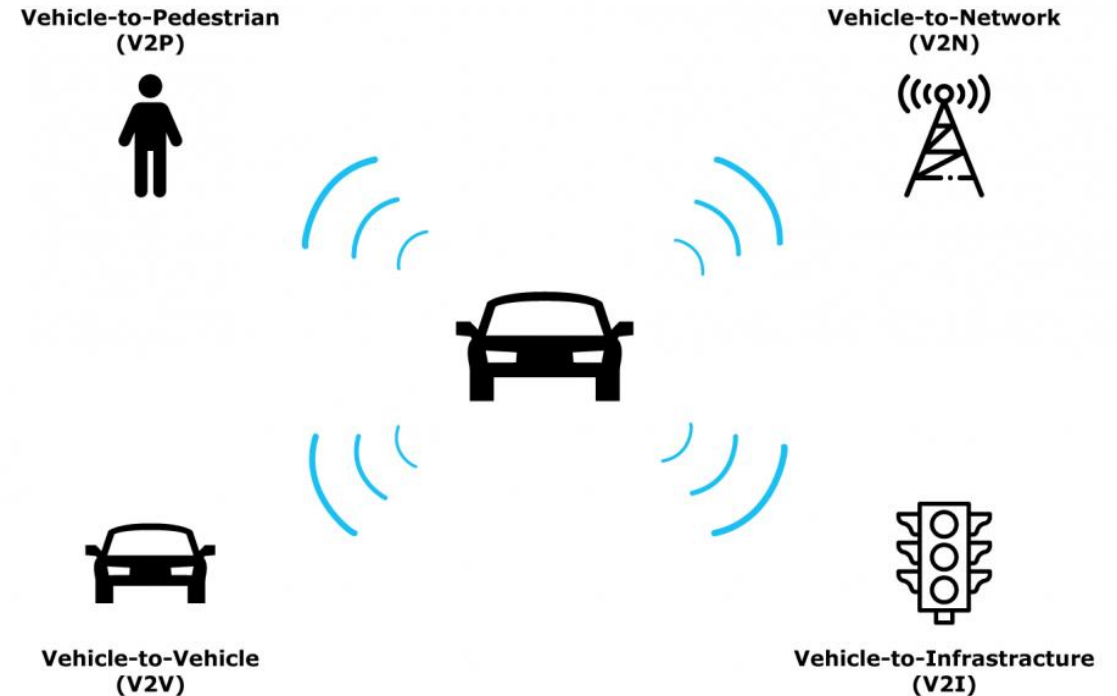
Butterfly Key Expansion

Why?

V2X Communication

Vehicle-to-Anything

- Cars broadcast signed messages to other cars and infrastructure.
 - Messages typically contain data like sender's position and speed.
 - Infrastructure like traffic lights can also send/receive messages.
- Signatures are authenticated using certificates.



Privacy issue: Mobility patterns of cars can be tracked via their signed V2X messages.

Solution: Pseudonym certificates with short-time credentials

V2X Security Credential Management System

SCMS

- Each end-user device has two types of certificates:
 - One long-term enrollment certificate
 - Identifies a valid device in the system
 - Used to request pseudonym certificates
 - Multiple short-lived pseudonym certificates
- SCMS relies on the following authorities:
 - Pseudonym Certificate Authority (PCA)
 - Responsible for issuing pseudonym certificates to devices
 - Registration Authority (RA)
 - Receives and validates requests for batches of pseudonym certificates from devices
 - Forwards pseudonym certificate requests to the PCA. Requests from different devices are shuffled together so that the PCA cannot link a set of requests to the same device.
 - Linkage Authority (LA) and Misuse Authority (MA) to handle revocation and misbehaviour.

SCMS Butterfly key expansion

The protocol

1. The vehicle generates two *caterpillar* key pairs:
 $\{s, S = s \cdot G\}$ for signatures and $\{e, E = e \cdot G\}$ for encryption.

The public *caterpillar* keys S and E are sent to the RA, together with two pseudorandom functions f_1, f_2 .

2. The RA uses the pseudorandom functions to generate a batch of public *cocoon* signature and encryption keys:
 - $\hat{S}_i = S + f_1(i) \cdot G$ for signatures and $\hat{E}_i = E + f_2(i) \cdot G$ for encryption
 - f_1 and f_2 are deterministic pre-defined methods for creating a pseudorandom integer mod l
 - Could for example be AES encryption of pre-defined inputs with a pre-shared key
3. The RA shuffles pairs of *cocoon* keys (\hat{S}_i, \hat{E}_i) from different vehicles together and send them individually to the PCA.

SCMS Butterfly key expansion

The protocol continued

4. The PCA receives a cocoon signature key \hat{S}_i , generates a random value r_i , and computes the device's *butterfly* public signature key $U_i = \hat{S}_i + r_i \cdot G$.

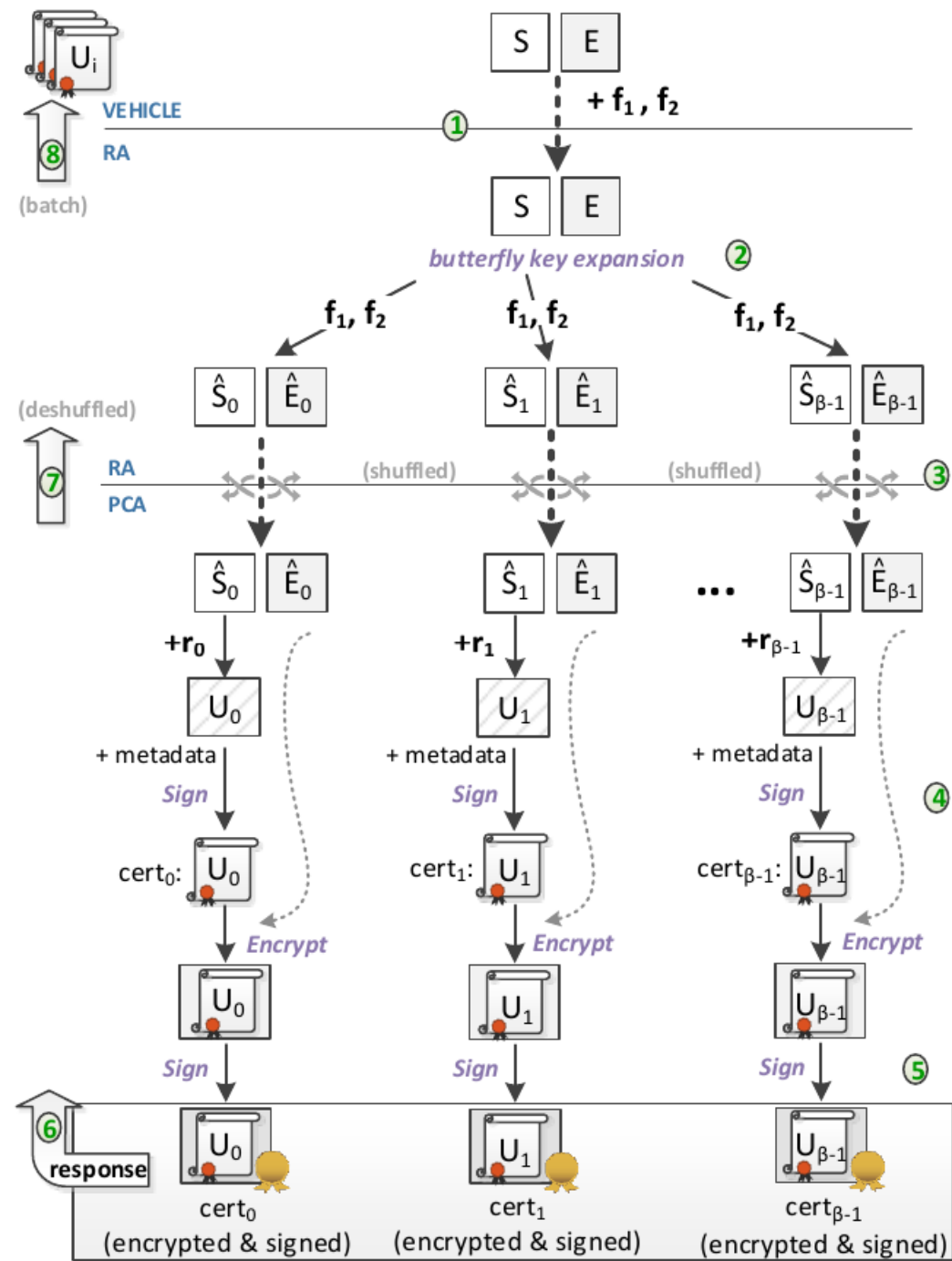
The PCA inserts U_i in a certificate, and signs it with its own private key.

The PCA uses the *cocoon* public encryption key \hat{E}_i to encrypt the certificate together with the random value r_i .

5. The PCA signs the encrypted response package.
6. The encrypted and signed response is sent back to the RA.
7. The RA deshuffles the requests and relays the PCA responses in batch to the requesting vehicles.
8. The requesting vehicle decrypts the package using its private *cocoon* encryption key $\hat{e}_i = e + f_2(i)$, and computes the corresponding *butterfly* private signature key $u_i = s + r_i + f_1(i)$

SCMS Butterfly Key Expansion

1. Device creates caterpillar keys and expansion functions.
2. RA generates cocoon keys.
3. RA shuffles certificate requests.
4. PCA generates butterfly keys and certificates and encrypts the response packages.
5. The PCA signs the encrypted responses.
6. Responses are returned to RA.
7. RA deshuffles responses and relays them to requesting device.
8. Device decrypts response and computes butterfly signing key.



Butterfly Key Expansion

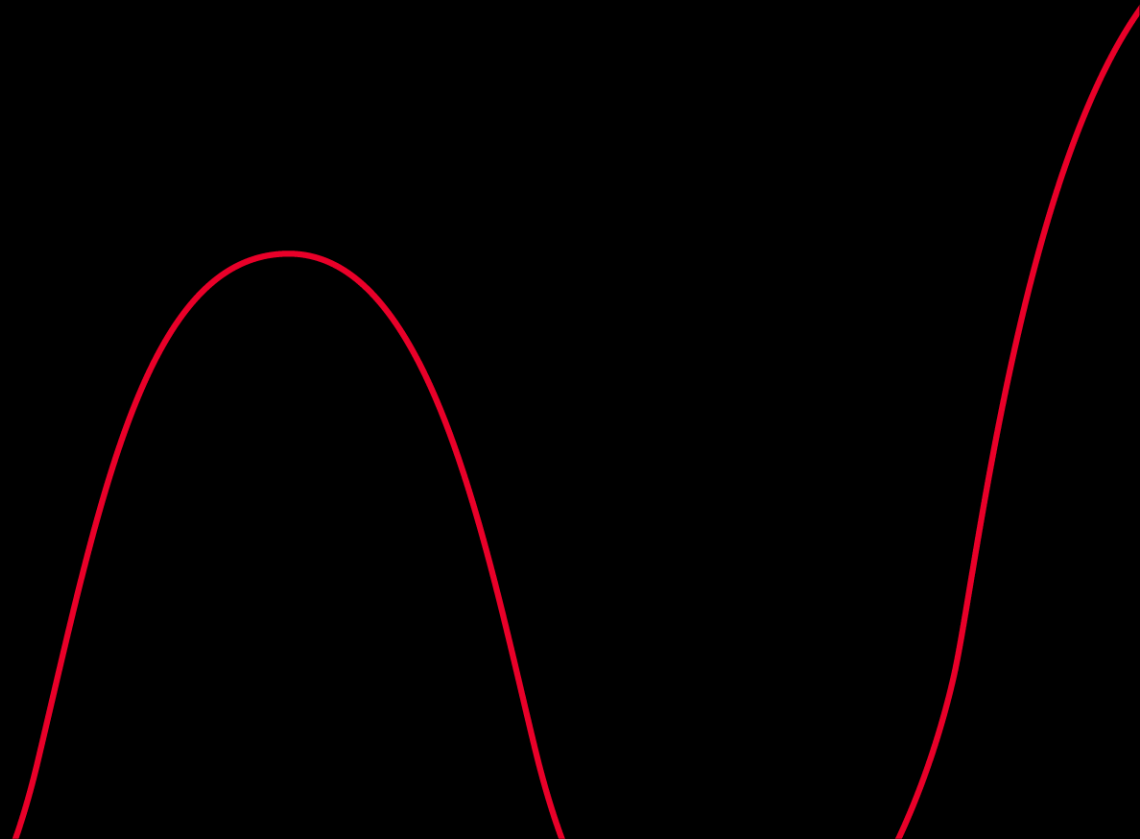
Some further points

- Resource efficiency:
 - The use of expansion functions allows RA/PCA to generate a large batch of certificates based on one single request from a device.
 - Batch creation can be done in off-peak hours.
- Unified Butterfly Key Expansion:
 - An optimized protocol which gets rid of the encryption key pairs.
 - Uses cocoon keys to encrypt PCA responses.
- Meddlers in the Middle attack for mixed approach:
 - If an (evil) RA promotes that it supports unified butterfly protocol, but it works with a PCA which uses the original butterfly protocol, then the RA can break the privacy and know which certificates belongs to which device.

Butterfly Key Expansion Resources

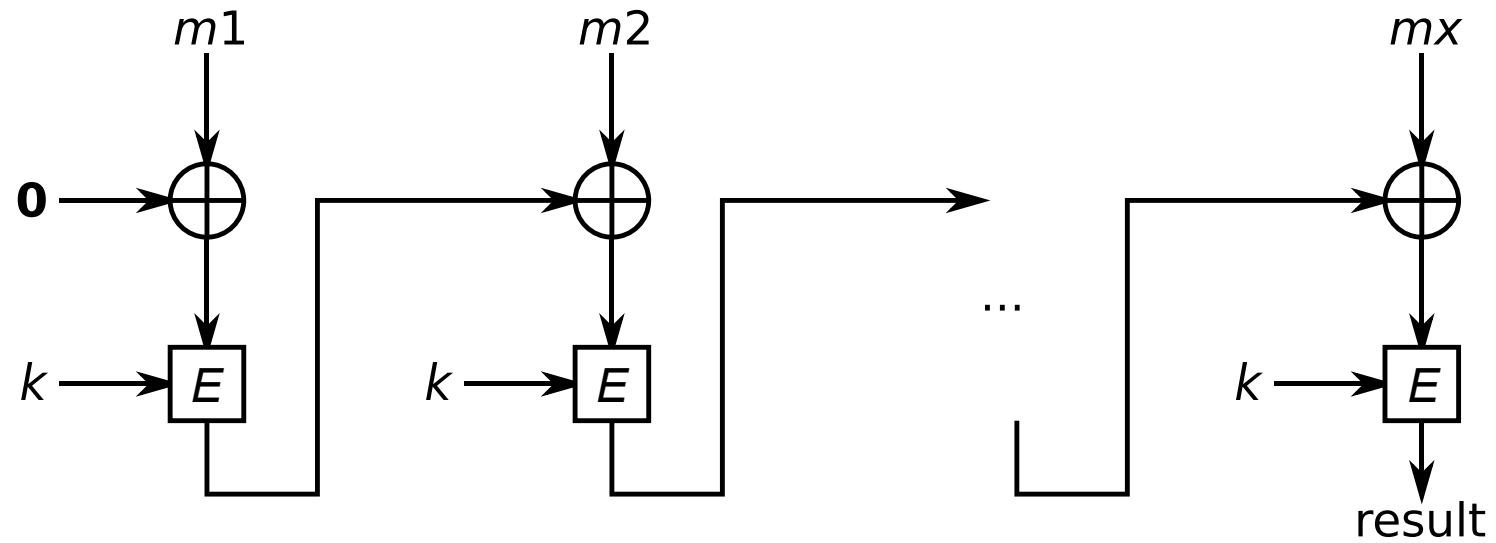
- Original Butterfly Key Expansion:
 - W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. A security credential management system for V2V communications. In *IEEE Vehicular Networking Conference (VNC'13)*, pages 1–8, 2013.
- Unified Butterfly Key Expansion:
 - Marcos A. Simplicio, Eduardo Lopes Cominetti, Harsh Kupwade Patil, Jefferson E. Ricardini, and Marcos Vinicius M. Silva. 2018. The Unified Butterfly Effect: Efficient Security Credential Management System for Vehicular Communications. In *2018 IEEE Vehicular Networking Conference (VNC)*. 1–8. <https://doi.org/10.1109/VNC.2018.8628369>
- Meddlers in the Middle for mixed mode:
 - https://www.youtube.com/watch?v=uhbKB_5u6jE
- Provable Security:
 - Alexandra Boldyreva, Virendra Kumar, Jiahao Sun. Provable Security Analysis of Butterfly Key Mechanism Protocol in IEEE 1609.2.1 Standard. In *IACR Cryptology ePrint Arch. (2024)*, 1674. <https://eprint.iacr.org/2024/1674>
- Butterfly Key Expansion with PQC:
 - Edward Eaton, Philippe Lamontagne, Peter Matsakis. Provably Secure Butterfly Key Expansion from the CRYSTALS Post-Quantum Schemes. In *IACR Cryptology ePrint Arch. (2024)*, 946. <https://eprint.iacr.org/2024/946>

Retail MAC



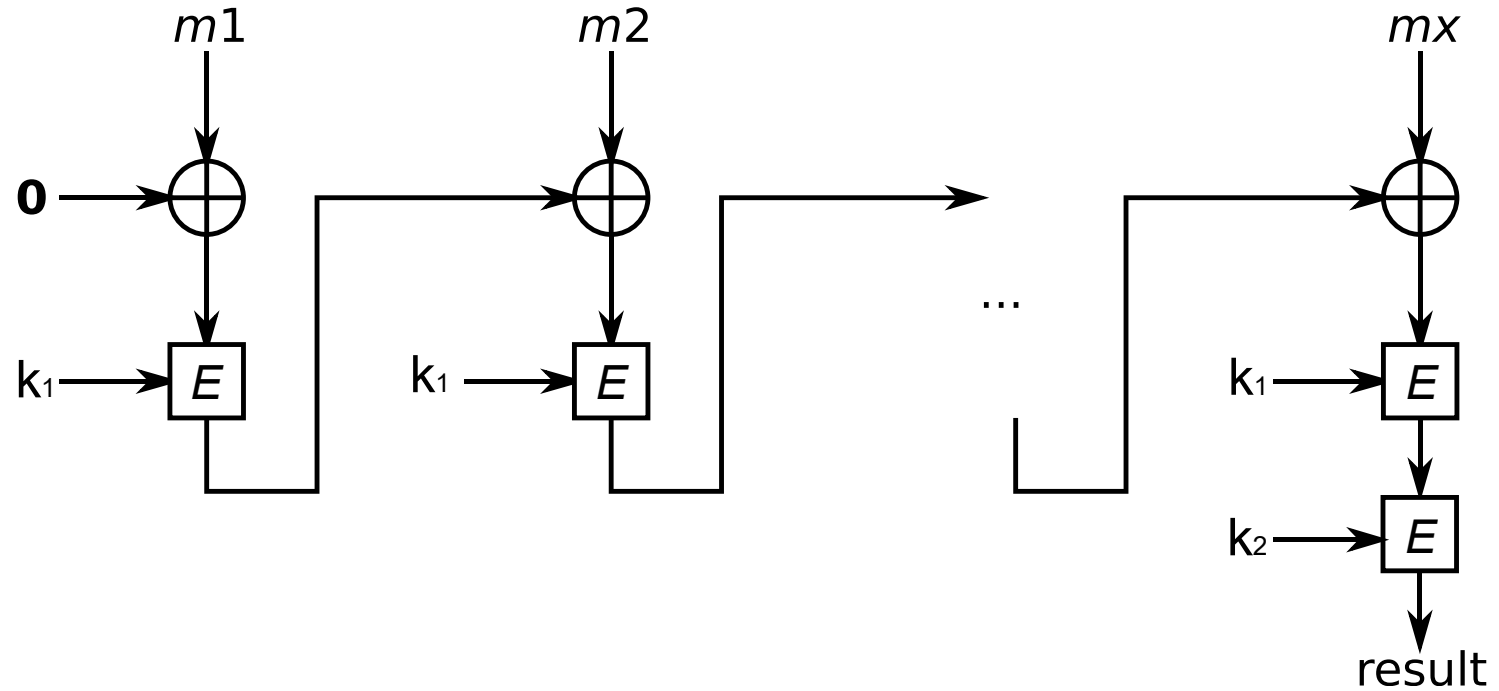
CBC-MAC

Not secure for variable-length messages



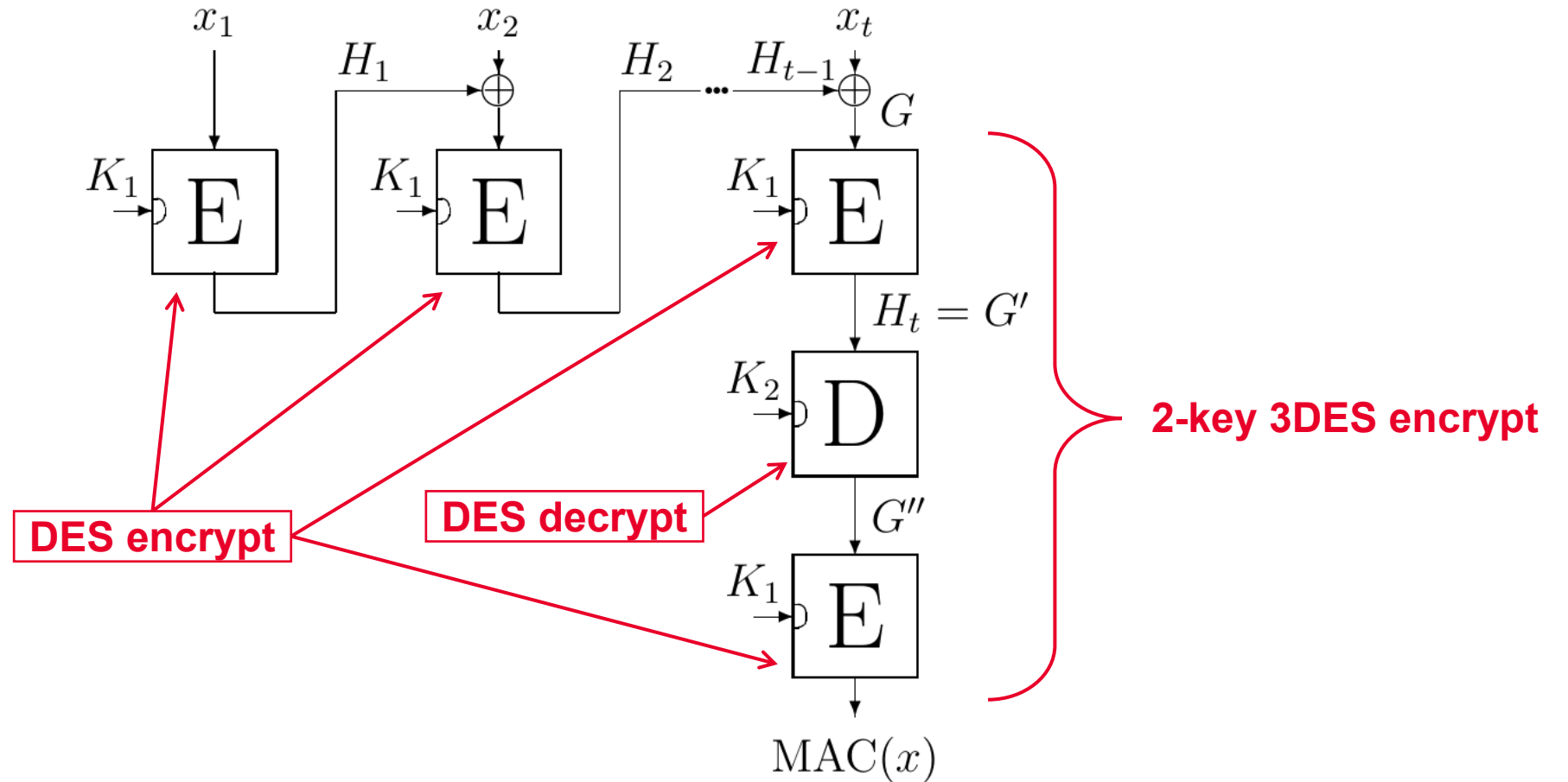
Encrypt-last-block CBC-MAC

Extends CBC-MAC to variable-length messages



Retail MAC

A special-case encrypt-last-block with 3DES



Retail MAC Resources

- Retail MAC is specified as MAC algorithm 3 in ISO/IEC 9797:
 - ISO/IEC 9797-1:2011 Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher
- Key recovery attack which requires $2^{32.5}$ known text-MAC pairs and $3 \cdot 2^{56}$ off-line computations to find the 112-bit key:
 - B. Preneel and P.C. van Oorschot. A key recovery attack on the ANSI X9.19 retail MAC. Originally appeared in *Electronics Letters* 32(17), pp. 1568–1569, 1996. Now available at <https://cosicdatabase.esat.kuleuven.be/backend/publications/files/journal/302>

Thank you

