Secret Can Be Public: Low-Memory AEAD Mode for High-Order Masking

Yusuke Naito (Mitsubishi Electric Corporation)

Yu Sasaki (NTT Social Informatics Laboratories)

Takeshi Sugawara (The University of Electro-Communications)

Wednesday, August 17, 10:00-11:40, Lotte Lehmann Hall
Session: Cryptanalysis II

Overview

1. HOMA: a new TBC-based AEAD mode
   - small memory size for high-order masking side-channel countermeasures
   - we protect only $s/2$ bits of the state, while we prove its security up to s bits.

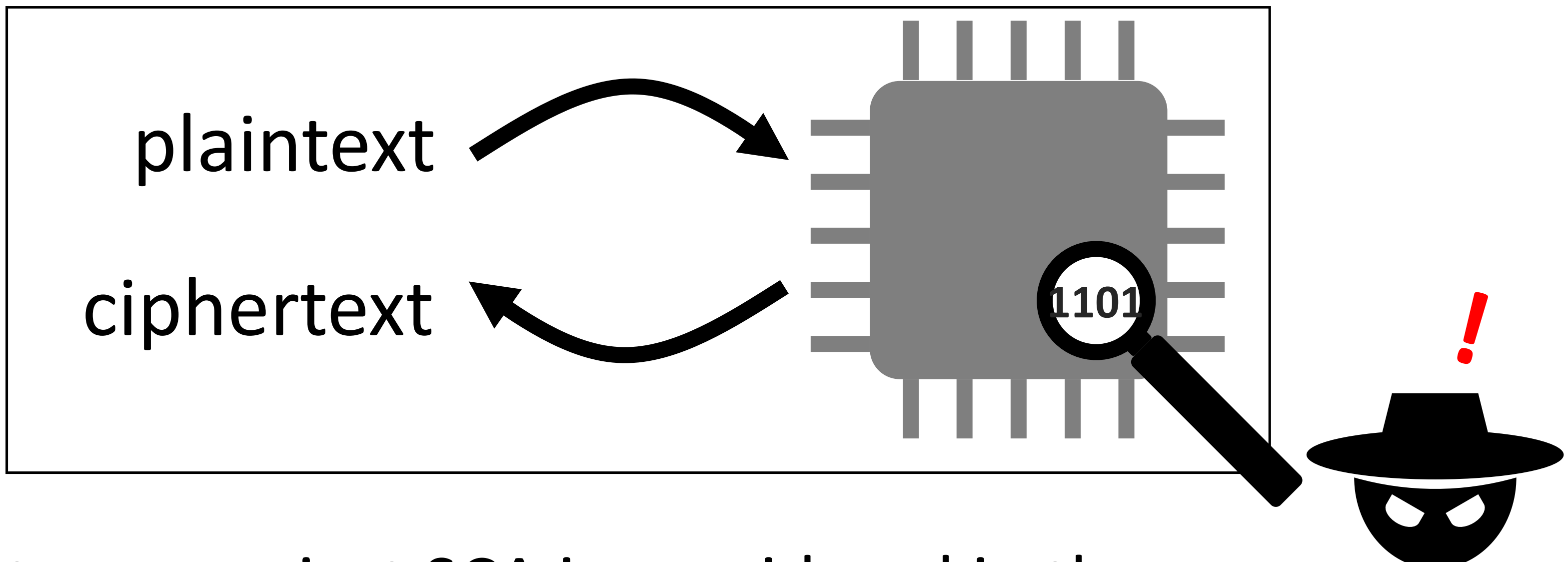
2. SKINNYee: a new SKINNY-based TBC instance
   - 64-bit block, 128-bit key, 259-bit tweak
   - Tweak and key should not be mixed in the schedule.

3. Hardware Implementation
   - Slightly bigger than state-of-the-art without masking.
   - Smallest for any protection order $d > 0$.

Side-Channel Analysis

- Side-channel analysis (SCA):

Besides the standard input/output of the function, the adversary steals some information from implementation features.



- Resistance against SCA is considered in the selection of future standards “lightweight cryptographic standardization process” by NIST.

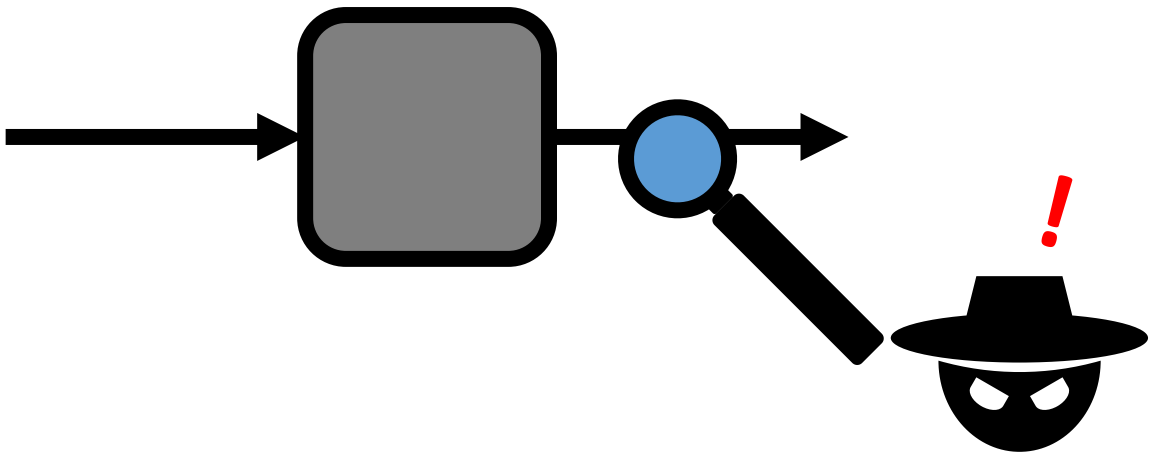
Directions on AEAD with SCA Protection

1. Leakage resilient
   - use leak-free component in a part of computations
   - minimize the use of such leak-free components
   - typically aims at optimizing the speed, not the size

2. Masking-friendly primitive
   - a primitive with a low multiplicative complexity
   - Mode-level optimization is not considered.

3. Low-memory AEAD mode
   - apply masking to all computations
   - minimize the memory size after the masking

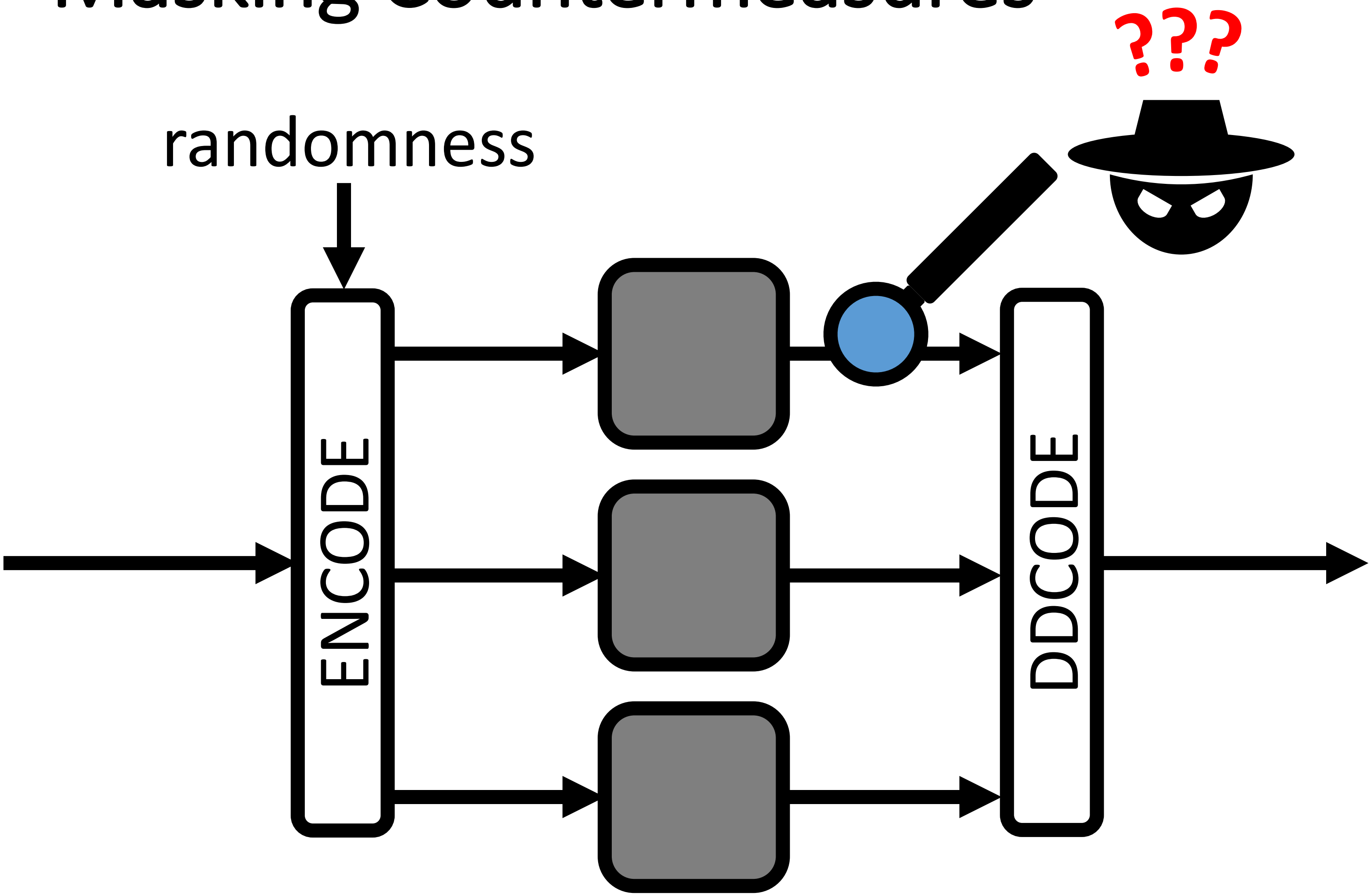
Side-Channel Adversary (Probing Model)



The (first-order) adversary probes a wire to get data.

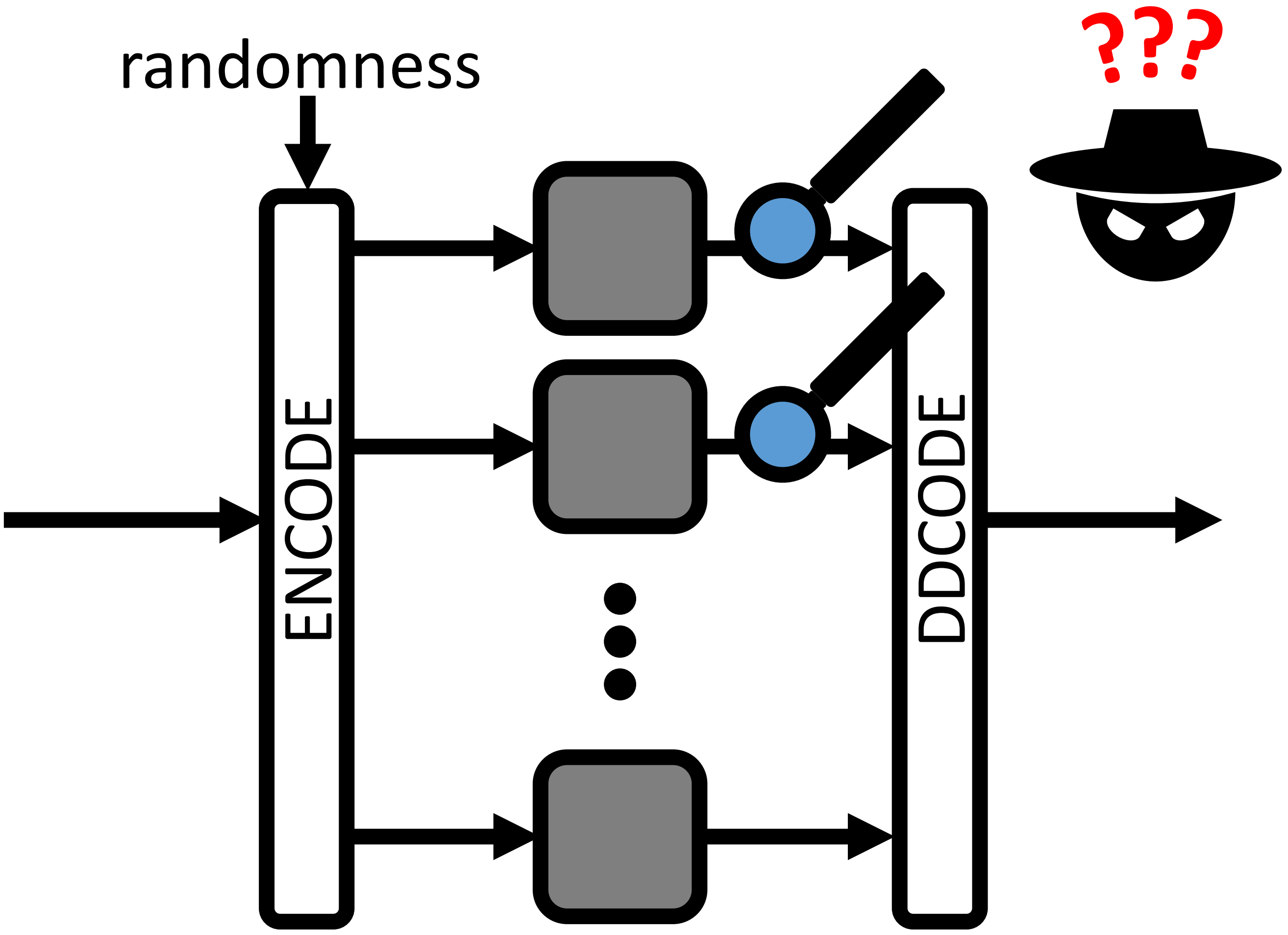
We assume the worst case scenario; the adversary fully gets the data on the wire.

Masking Countermeasures



Data is encoded to multiple shares.
Unable to get data only by probing a wire.

High-Order Masking



Powerful adversary may probe d wires. $(d > 1)$
Such adversaries can be avoided by making more shares.

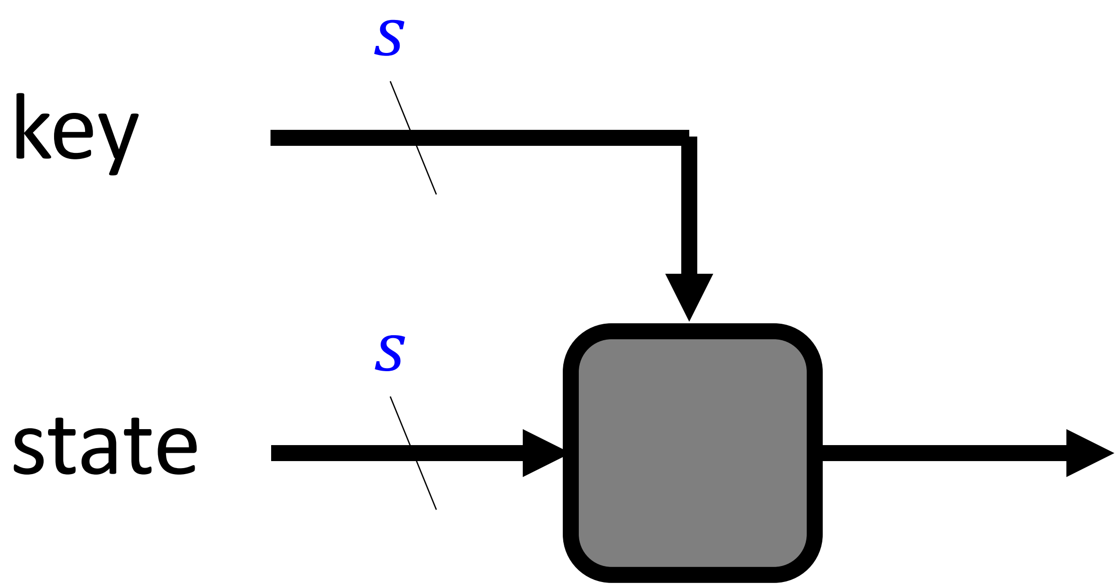
Research Motivation

- Large memory overhead for multiple shares particularly for a high-order (large d).

- $d + 1$ masking schemes encode a state into $d + 1$ shares.

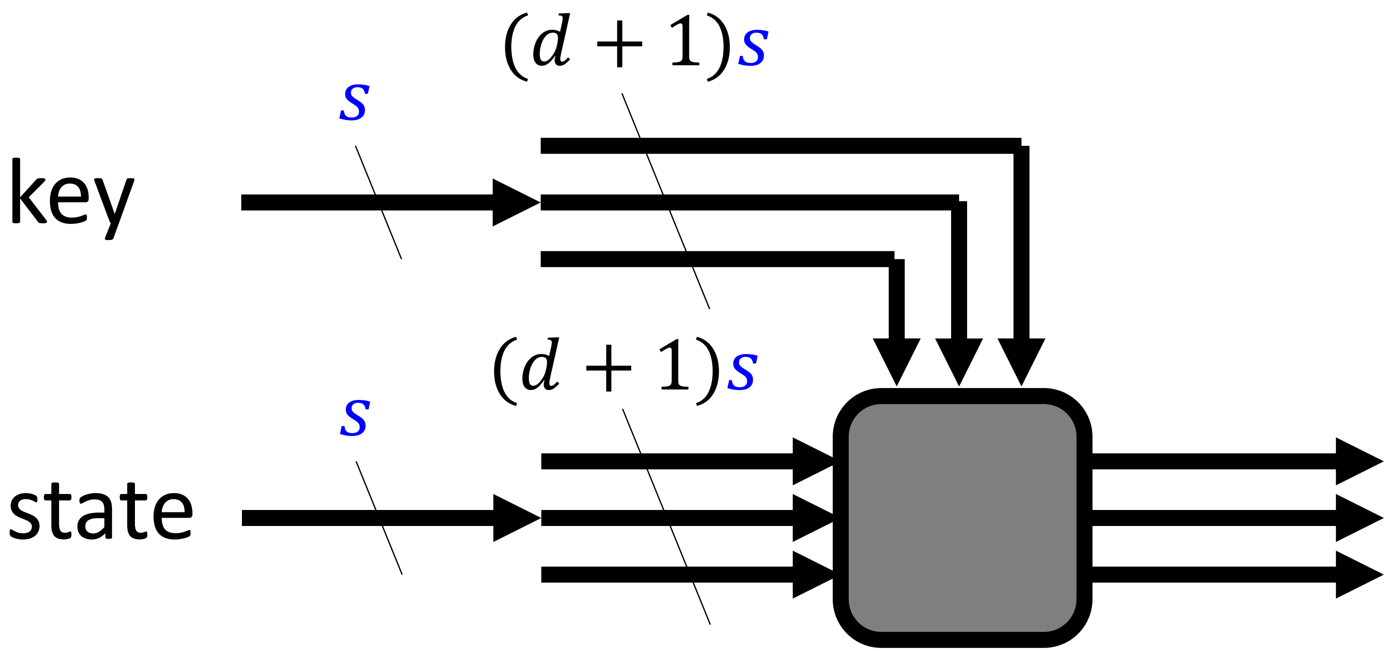
- We need a new design that achieves a small memory size after a masking with protection order d.

Minimum State Size



- To achieve s-bit security, the state size and the key size must be at least s bits.
- Otherwise, the key or state can be guessed with a complexity less than 2^s.

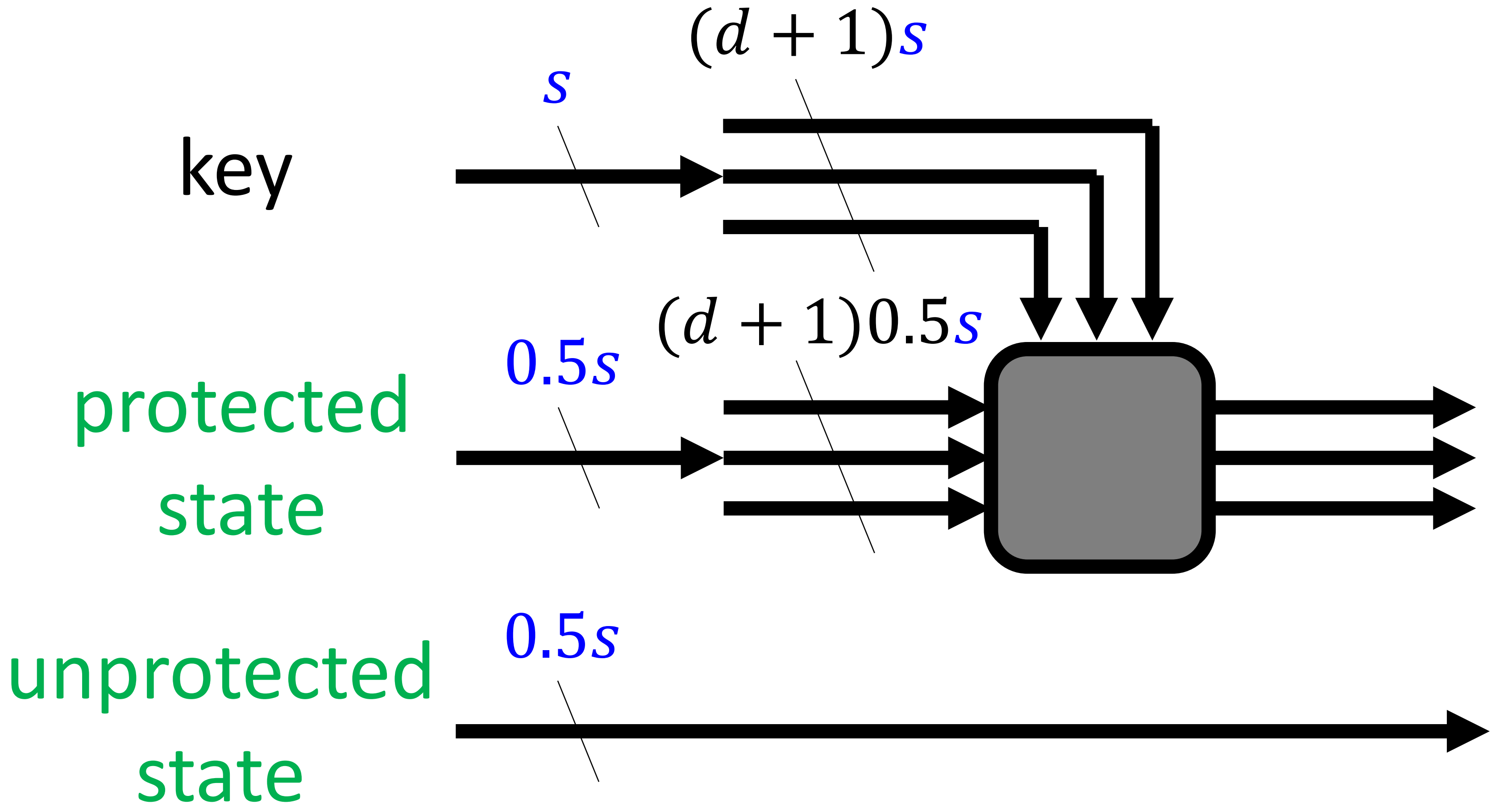
Folklore on the Memory Size for Masking



Folklore

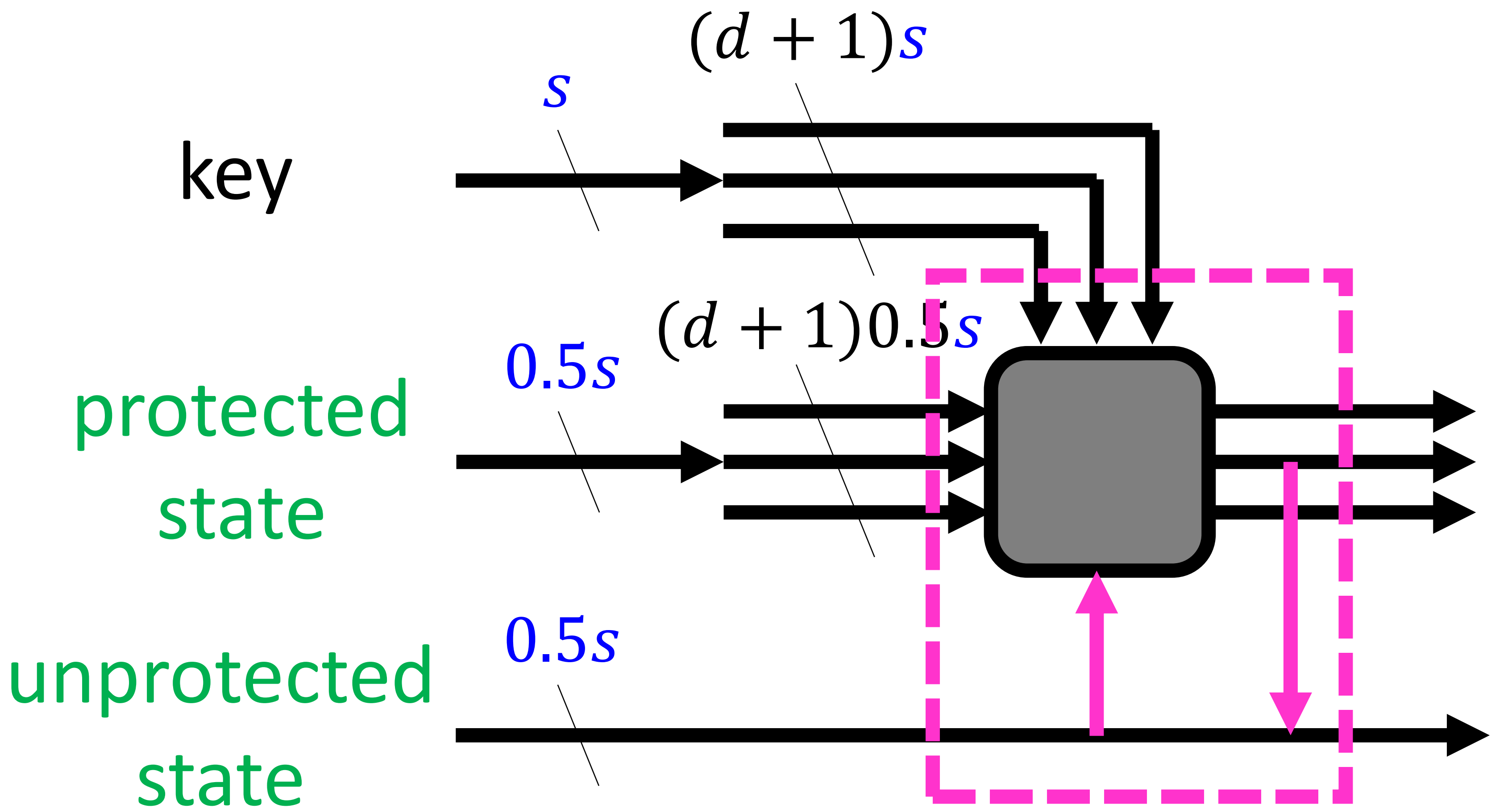
By encoding the state and the key into d+1 shares, the total memory size is at least $(d+1)2s$ *bits*

Overview of Our Approach 1



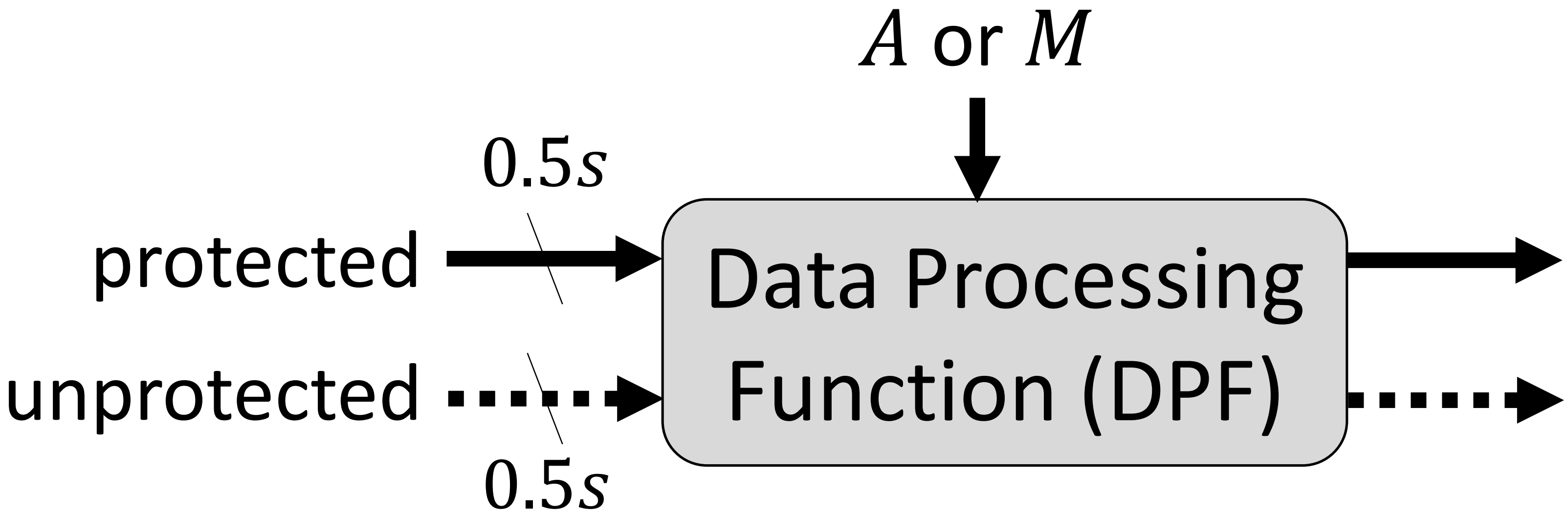
1. Leave $s/2$ bits "*unprotected*". Asymptotically achieves $(d+1)1.5s$ bits of memory.

Overview of Our Approach 2



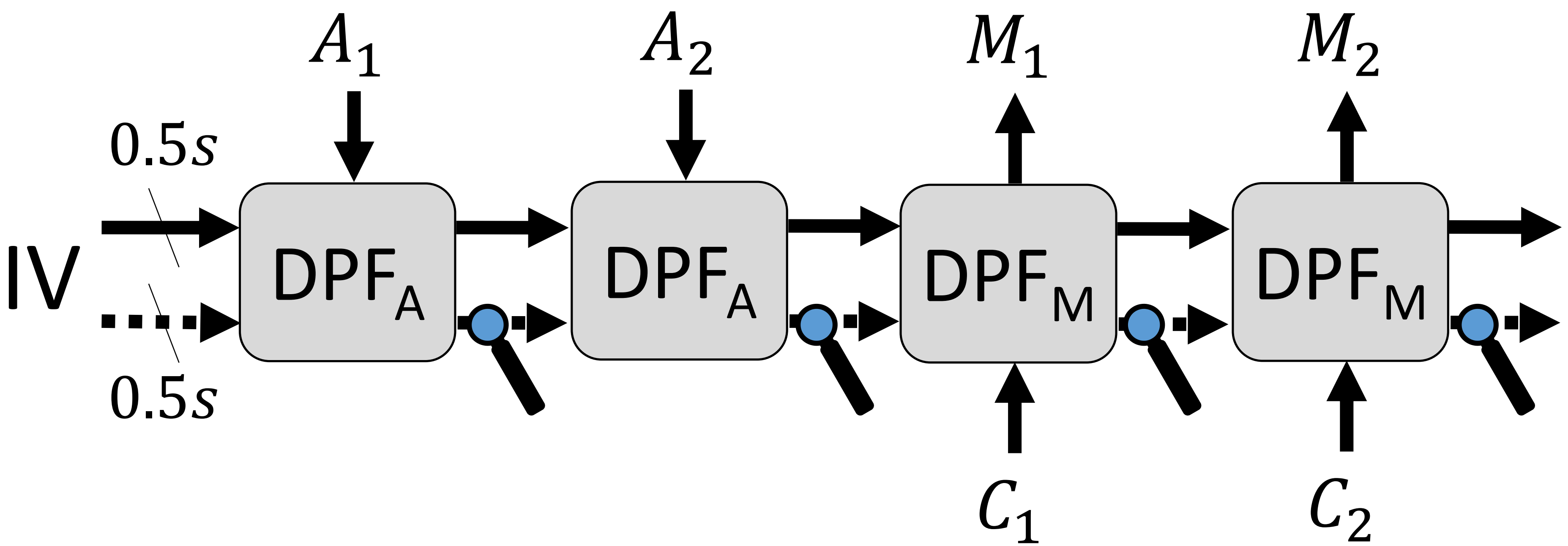
2. Devise new operations to securely mix protected and unprotected values.

General Description



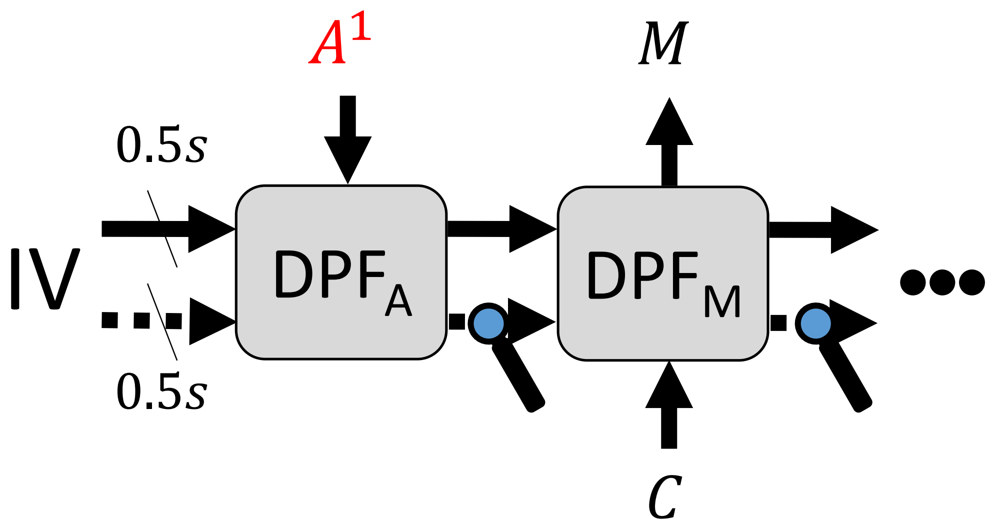
- With a standard nonce-based AEAD, the construction is generally attacked with $2^{0.5s}$.

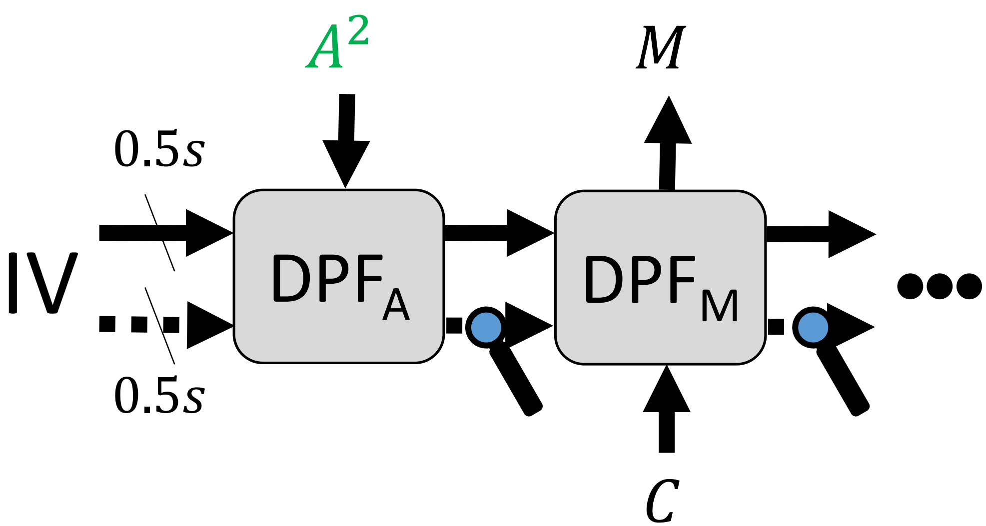
Decryption with Unprotected State



- For any decryption query, unprotected values are leaked even with an invalid tag.
- The **verify-then-decrypt** policy cannot stand against SCA adversaries.

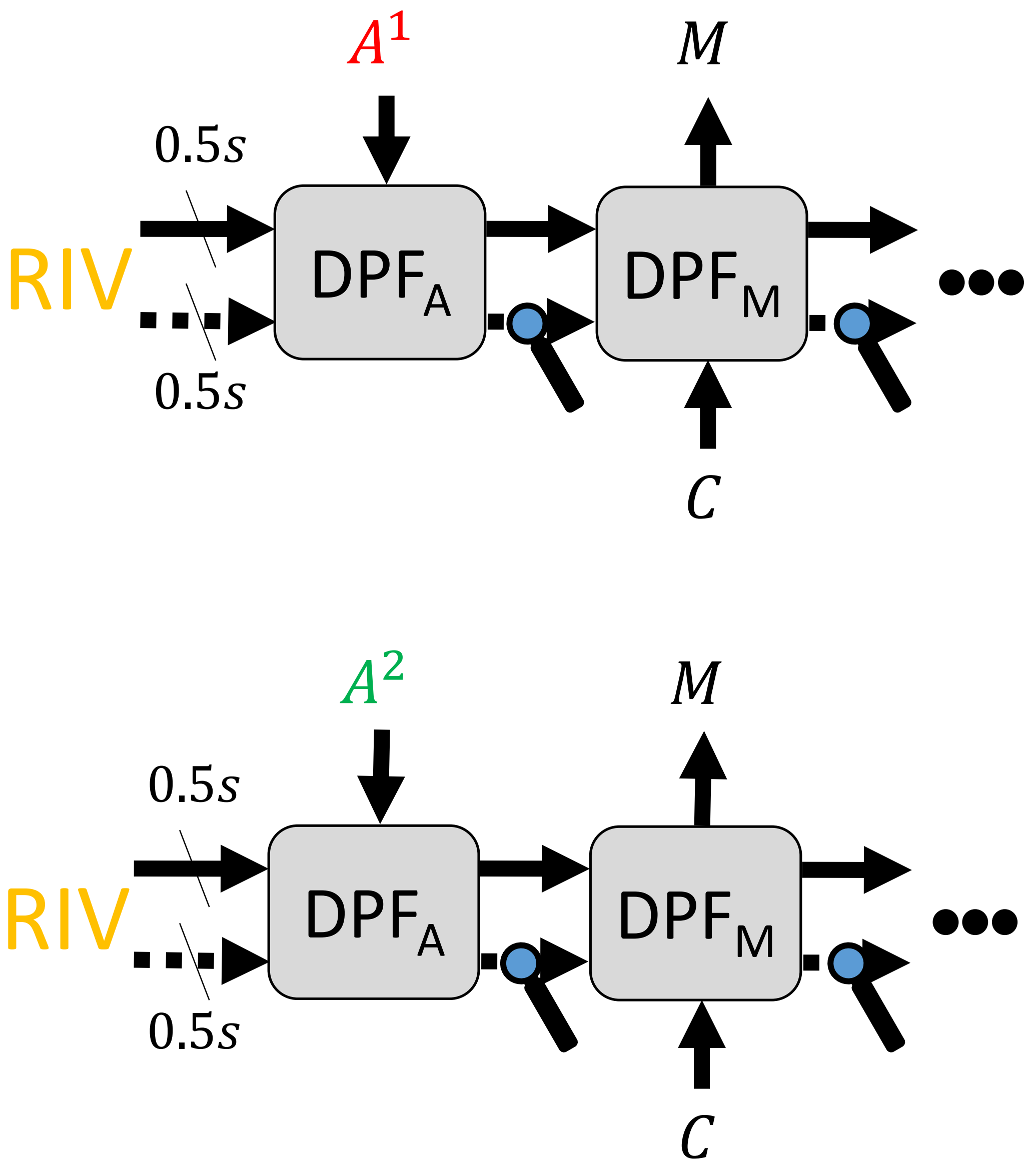
An Attack for Fixed IV





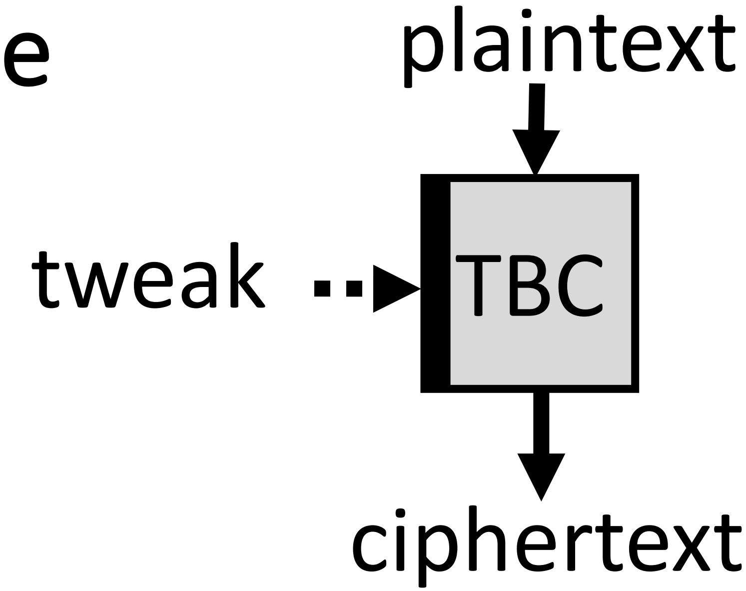
- Make $2^{0.5s}$ Dec queries (N, A, C, T) to get unprotected values for various A.
- Find (A^1, A^2) colliding in unprotected values.
- Make an Enc query (N, A^1, M^*) for any M^* to get (C^*, T^*).
- (N, A^2, C^*, T^*) is a valid pair.

Use of Random IV



- We force IV to be randomly determined for each Enc query.
- Adversaries can no longer play with Dec oracle before IV is determined. (otherwise, random IV needs to be guessed.)

Primitive Choice



Plaintext/ciphertext: directly updated by a key, thus needs protection.

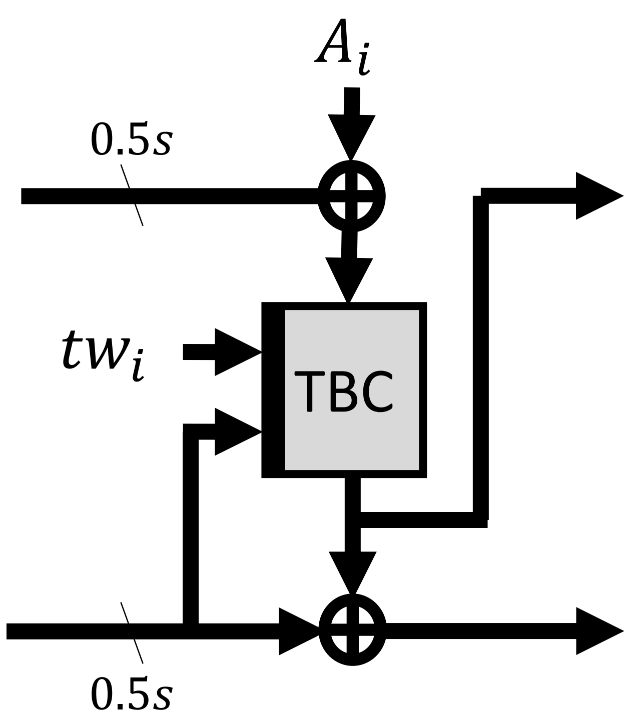
- ➢ Protected state is assigned to plaintext.

Tweak: a public value, thus no need of protection.

- ➢ Unprotected state and other public data (nonce, ctr, data input) are assigned to tweak.

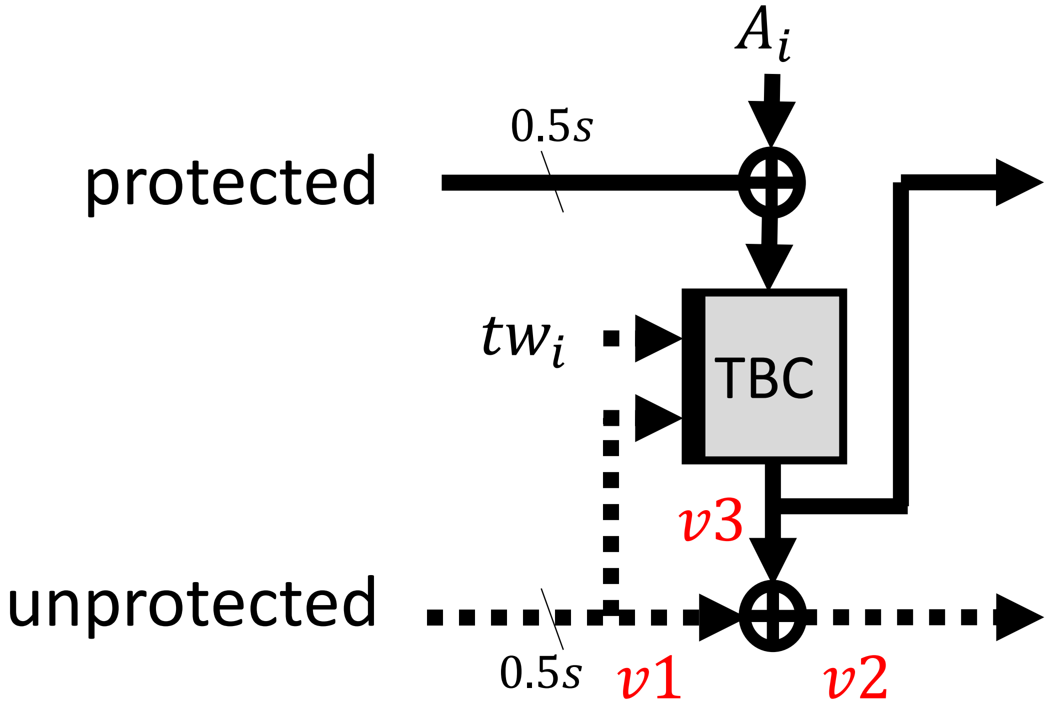
BBB Security of PFB_plus [NSS,EC20]

- HOMA applies the protection only for the plaintext-ciphertext of TBC with $0.5s$-bit block.
- The idea of PFB_plus helps to ensure s-bit security by using $0.5s$-bit block TBC.



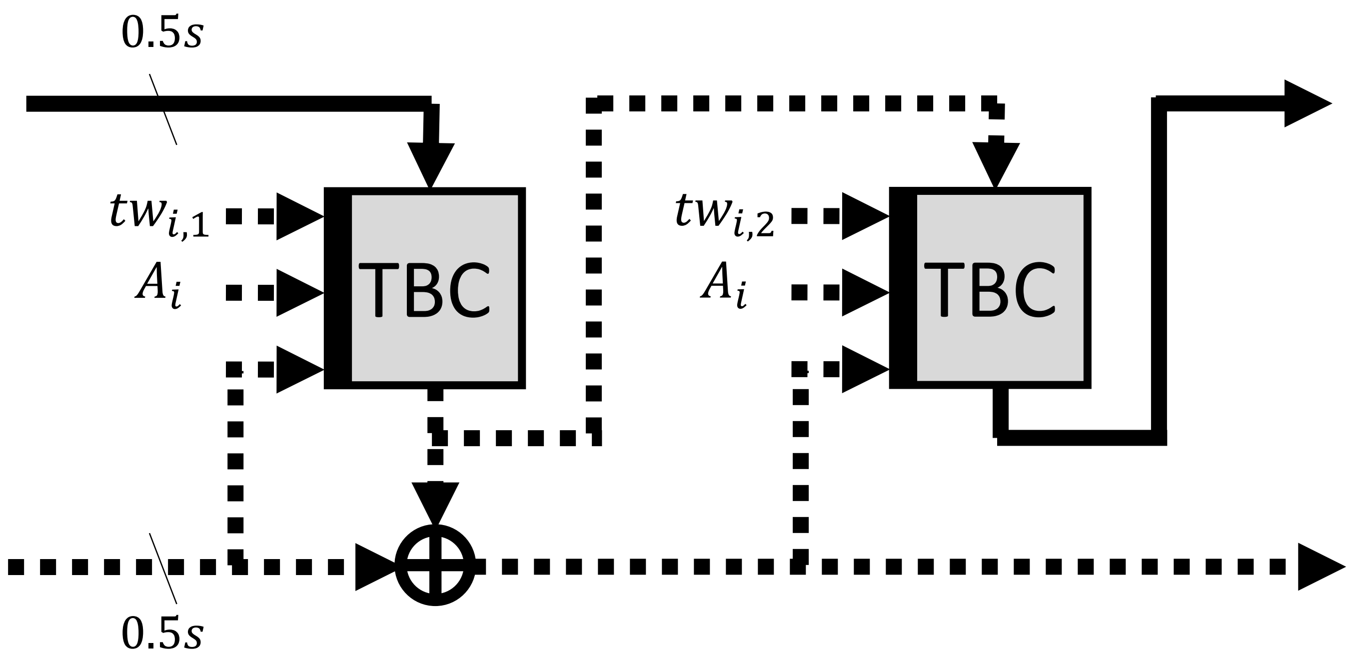
PFB_plus is Broken If Unprotected

By unprotecting a half of the state of PFB_plus, the construction is broken only by $2^{0.5s}$.



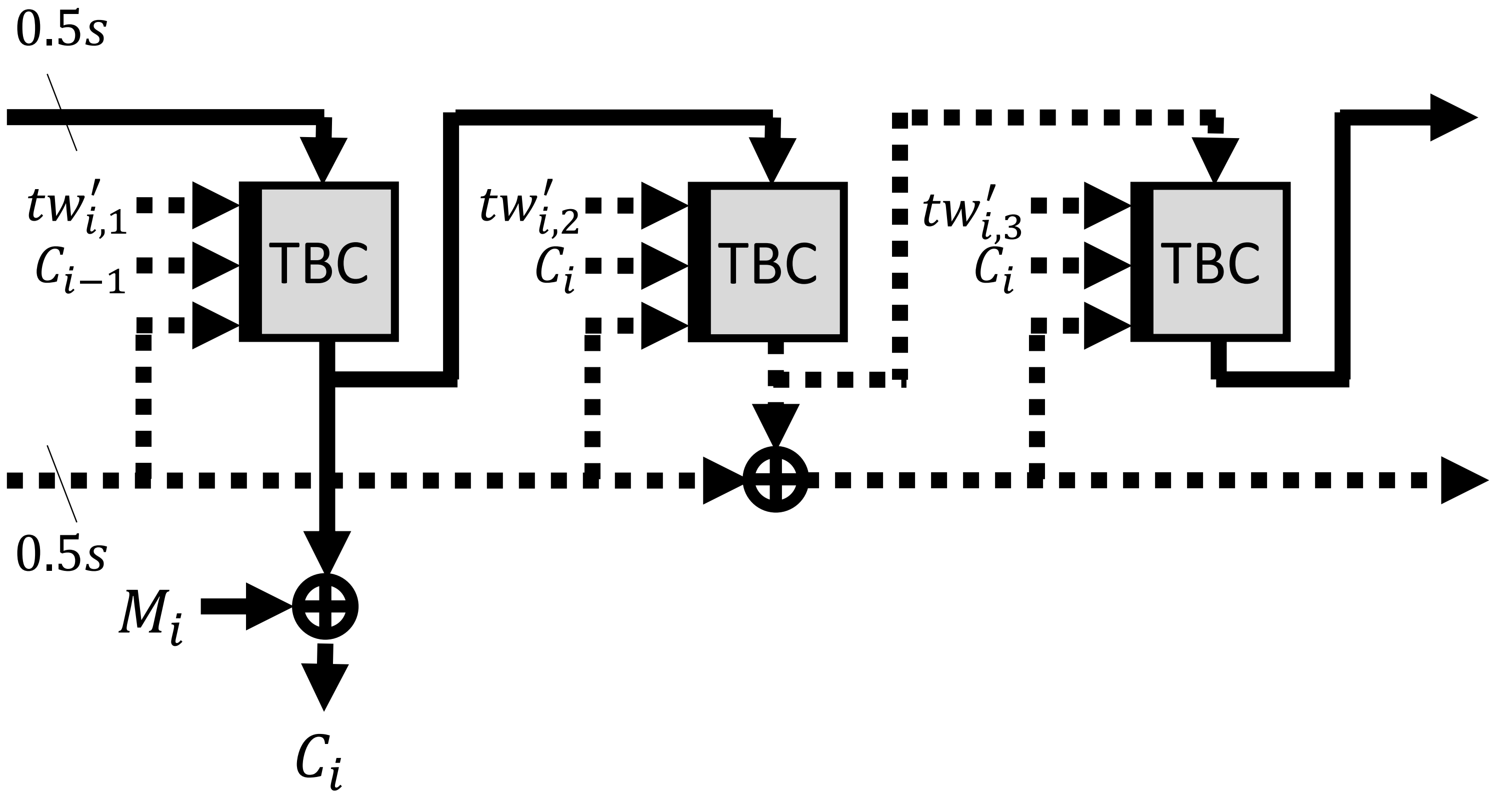
The protected value $v3$ is recovered from unprotected values $v1$ and $v2$.

Overview of DPF_A



- A TBC-call generates $0.5s$-bit unpredictable value. To mix the s-bit state, 2 TBC calls are needed.
- Compared to PFB_plus:
  - bigger tweak ➡ more memory for $d = 0$.
  - smaller protected state ➡ less memory for $d > 0$.

Overview of DPF_M



- For DPF_M, the first TBC generates a key stream, and other 2 TBC calls mix the state.

Security Proof Overview

- Strong tweakable PRP (STPRP) assumption for the underlying TBC with $0.5s$-bit block.
- s-bit security is proved.

Intuition of authenticity

- For each DPF, we ensure that s-bit unpredictable value is produced.

Intuition of privacy

- Independence of each TBC invocation is ensured by the nonce and the counter.

New TBC: SKINNYee

- For **128**-bit security, HOMA needs a TBC with **64**-bit block, **128**-bit key, and **259**-bit tweak.

  ➡ No exiting TBC supports those parameters.

- “Unprotected values (tweak)” and “protected values (key)” defined by the mode must not be mixed inside the primitive.

  ➡ “Tweakey” framework is useful.

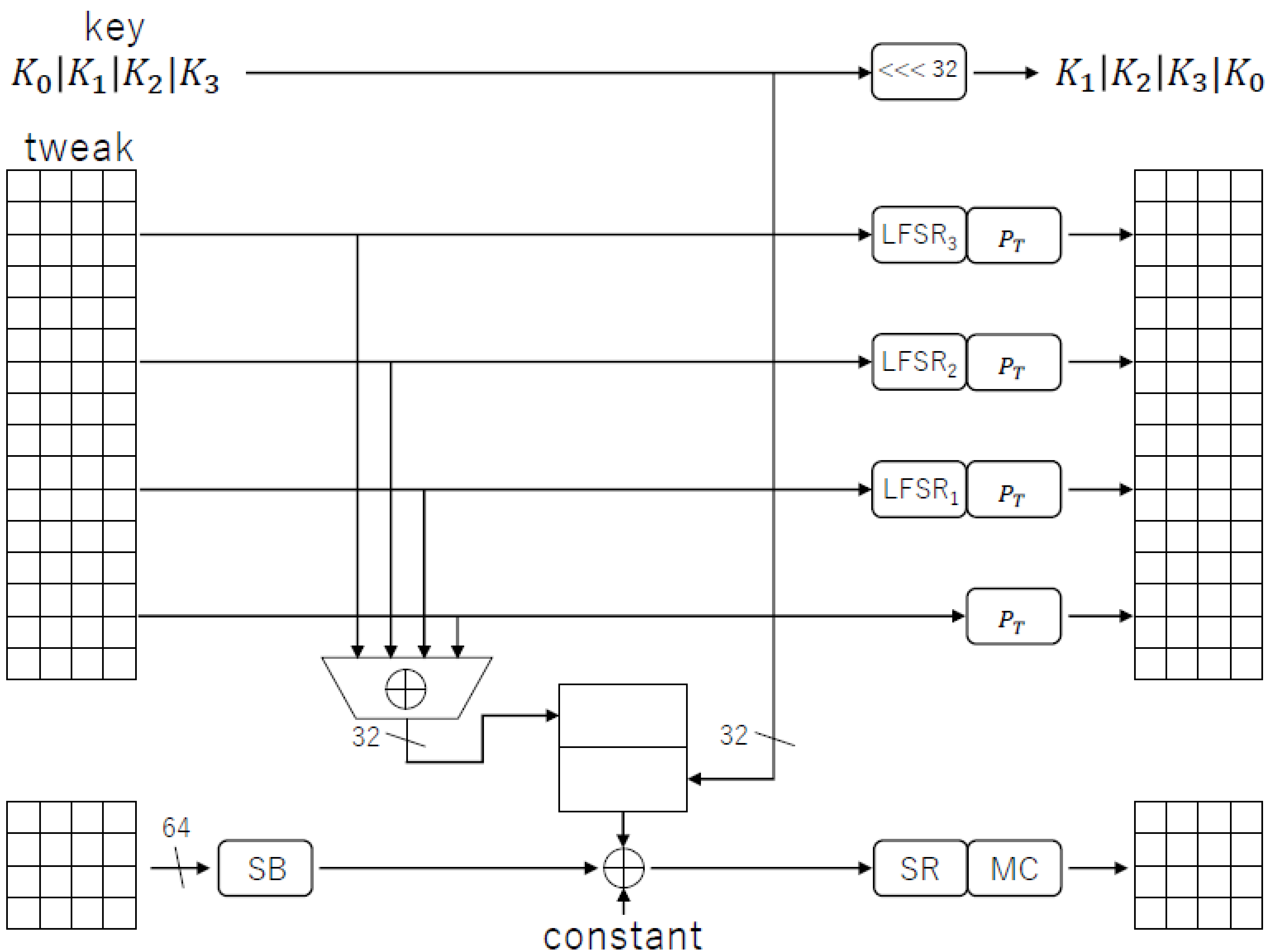
- **387**-bit tweakey is too large for 64-bit block (TK7). (No efficient way exists to support TK7)

Design Features

- Tweakey supports variable tweak and key sizes.
- This is not important for HOMA so we drop it:
  - use TK4 of SKINNYe to handle 256-bit tweak
  - inject key to the lower half of the state.
  - MILP ensures limited number of active S-boxes.

- The remaining 3-bit tweak is processed by the *elastic-tweak* [CDJMNS,Indocrypt2019], but we improve it to achieve a smaller memory.

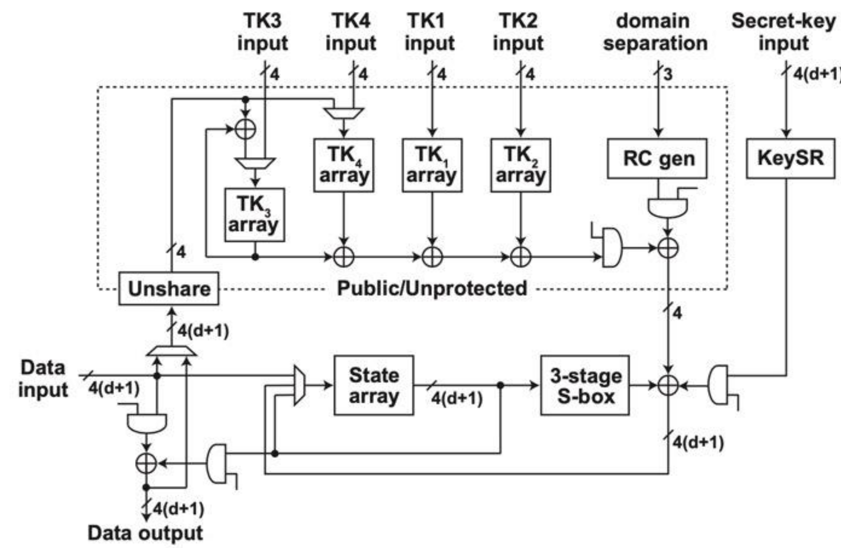
A Sketch of Round Function



Implementation Features

- ASIC hardware performance evaluation with HPC2* masking scheme for $d \in \{0, \dots, 5\}$

- Comparison with PFB_Plus with the same impl. policy



*Cassiers, G., Gregoire, B., Levi, I., Standaert, F.X.: Hardware Private Circuits: From Trivial Composition to Full Verification. IEEE Transactions on Computers pp. 1–1 (2020)

Implementation Results with SKINNY Variants

Table 3. Hardware performances in gate equivalent (GE) for $d \in \{0, \cdots, 5\}$

Component	HOMA						PFB_Plus					
	$d = 0$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 0$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Total	4,981	6,283	8,226	10,392	12,782	15,487	4,569	6,884	9,667	12,675	15,941	19,724
S-box	161	501	1,087	1,897	2,931	4,189	161	501	1,087	1,897	2,931	4,189
State array	542	1,046	1,573	2,097	2,621	3,240	540	1,049	1,571	2,094	2,619	3,238
TK_1 array	636	549	549	549	549	549	637	1,231	1,845	2,459	3,083	3,818
TK_2 array	844	749	744	748	744	748	674	1,296	1,938	2,578	3,239	3,989
TK_3 array	675	585	586	585	585	586	746	656	657	656	656	656
TK_4 array	675	577	576	577	577	576	865	782	782	780	780	781
KeySR	735	1,468	2,201	2,935	3,668	4,402	—	—	—	—	—	—
Shift reg.	—	—	—	—	—	—	377	754	1,131	1,508	1,885	2,262

- **HOMA**'s memory size is bigger than **PFB_Plus** for implementations without SCA protection $d = 0$.
- HOMA is advantageous for any $d > 0$.
- The Improved factor is bigger than the S-box size.

➡ Our results cannot be reached by improving S-box.

Concluding Remarks

- We proposed a new TBC-based AEAD mode HOMA, which achieves small memory for high-order masking.
- Our hardware implementations show that HOMA with our SKINNY-based variants is
  - slightly bigger than state-of-the-art without masking, and
  - smallest for any protection order $d > 0$.

Future Work

- New modes to ensure s-bit security based on a TBC with a smaller block size than $0.5s$ bits, along with a specific TBC design to support such configuration.

Thank you for your attention!!