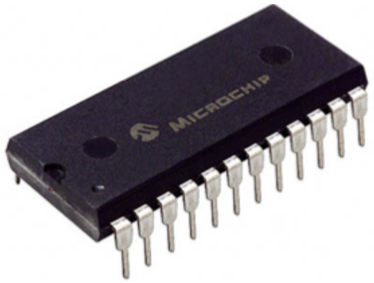
Robustly Reusable Fuzzy Extractor from Standard Assumptions

Yunhua Wen and Shengli Liu

Shanghai Jiao Tong University

Problem

- Randomness is crucial in cryptography (e.g. sk).
- However, uniformly distributed and accurately reproducible string is rare in practice.
- There are many imperfect random sources, e.g.

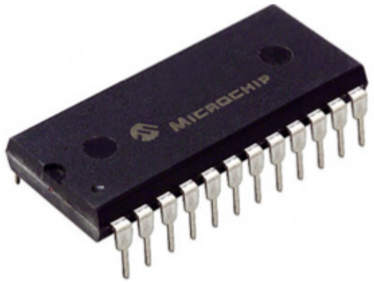


Physically Unclonable Functions (PUFs)

Biometric Information

Problem

- Randomness is crucial in cryptography (e.g. sk).
- However, uniformly distributed and accurately reproducible string is rare in practice.
- There are many imperfect random sources, e.g.



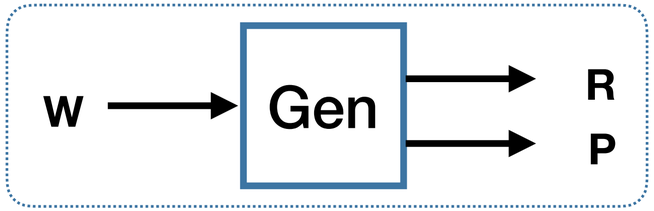
Physically Unclonable Functions (PUFs)

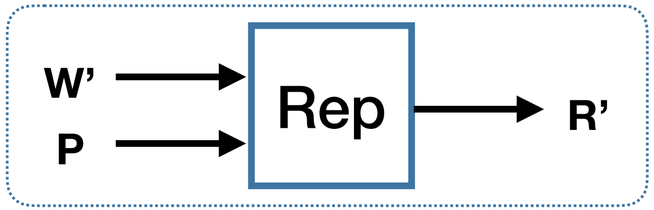
Biometric Information

Problem: How to use such imperfect random sources in cryptography?

Fuzzy Extractor

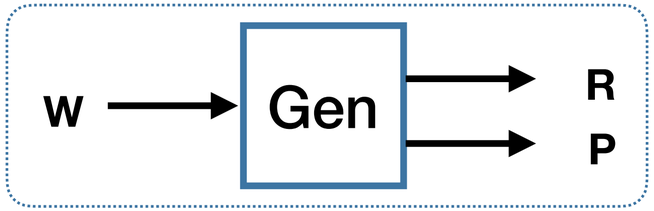
- **Gen(w)**
  - **Input**: a weak random secret w.
  - **Output**: the extracted key R and a public helper string P.
- **Rep(w', P)**
  - **Input**: a noisy version w' and the public helper string P.
  - **Output**: the extracted key R'.

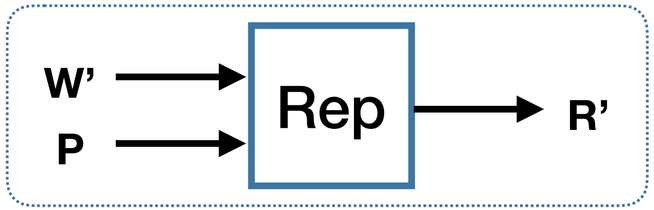




Fuzzy Extractor

- **Gen(w)**
  - **Input**: a weak random secret w.
  - **Output**: the extracted key R and a public helper string P.
- **Rep(w', P)**
  - **Input**: a noisy version w' and the public helper string P.
  - **Output**: the extracted key R'.



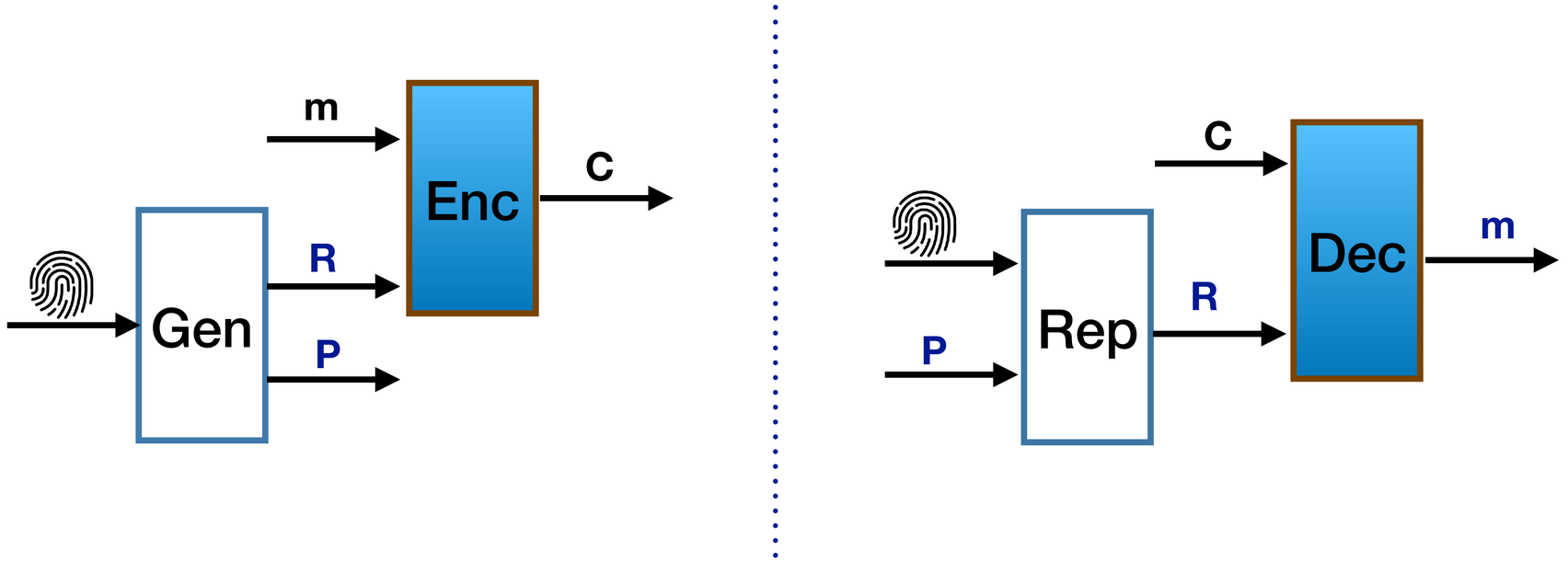


Correctness: If w' is close enough to w, R'=R.

Security: R is pseudorandom given P.

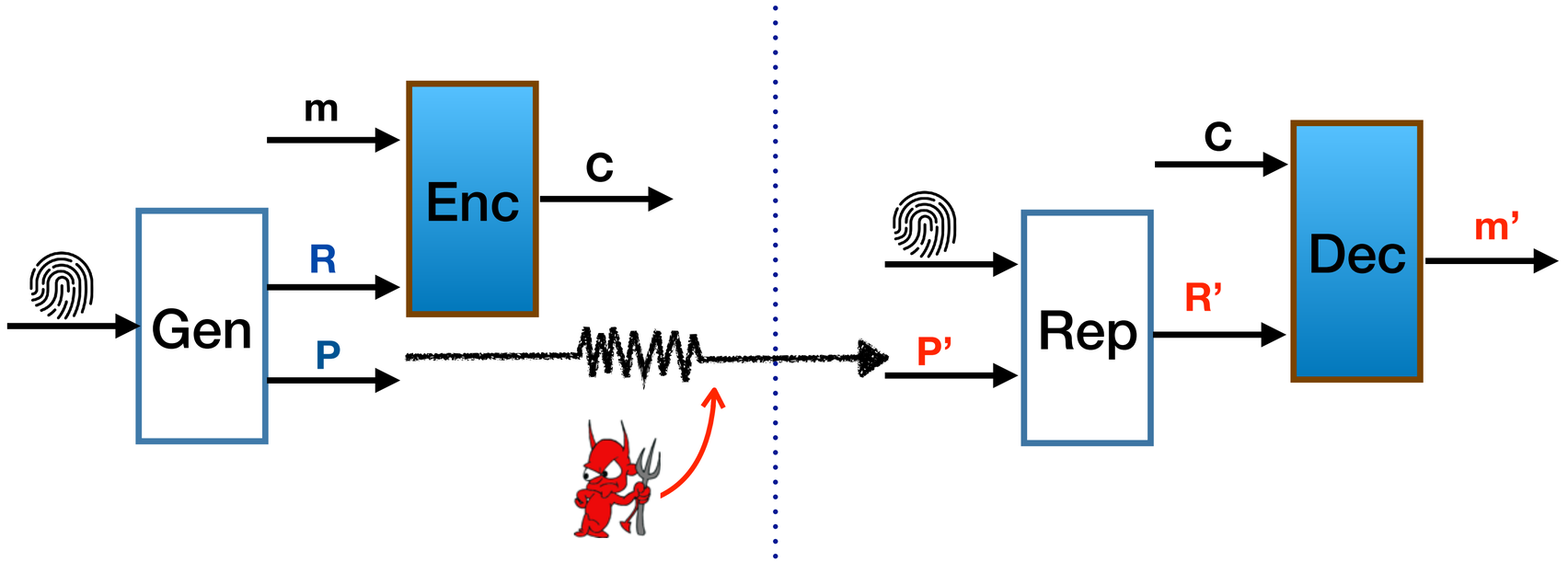
Applications

Application in **Encryption** and **Decryption**:



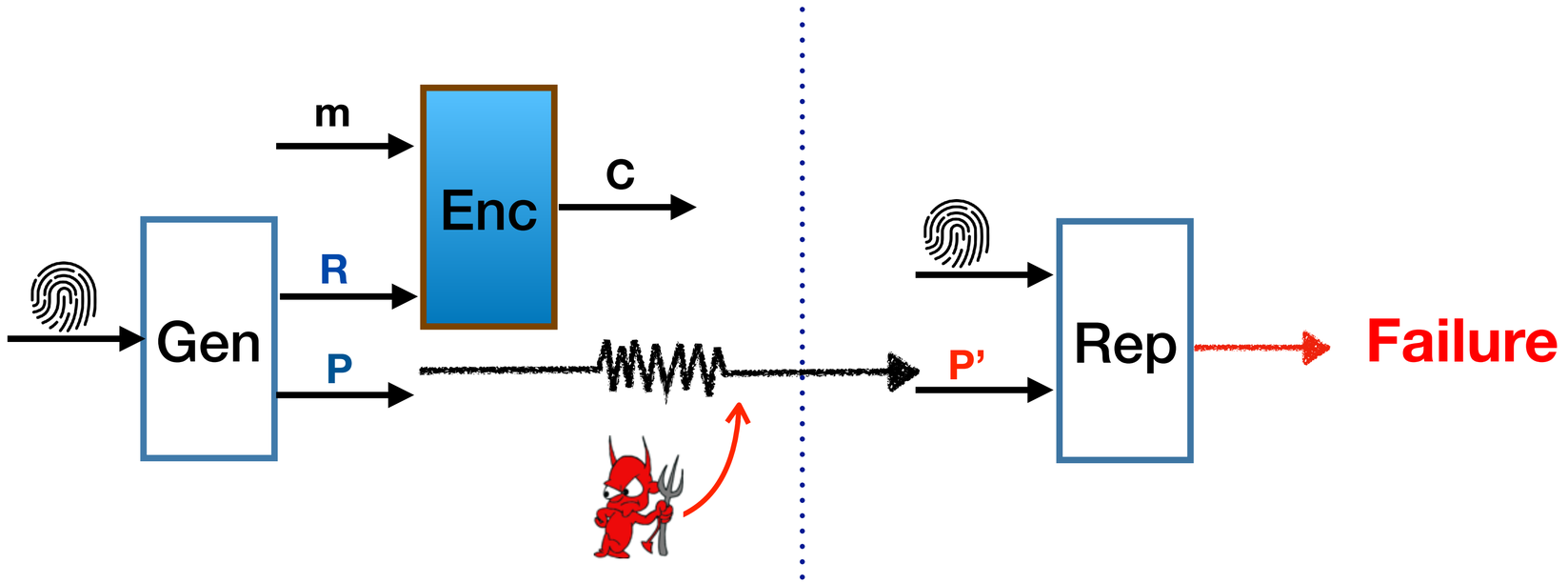
Users do not need to store the secret key R.

Robust Fuzzy Extractor



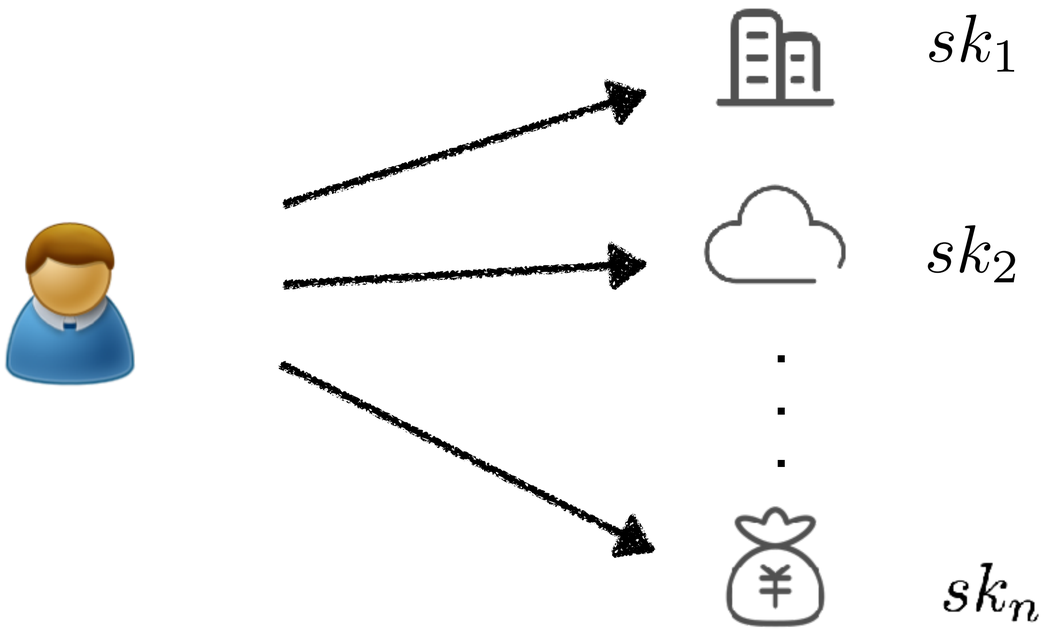
The user may get a wrong key R' without notifications.

Robust Fuzzy Extractor



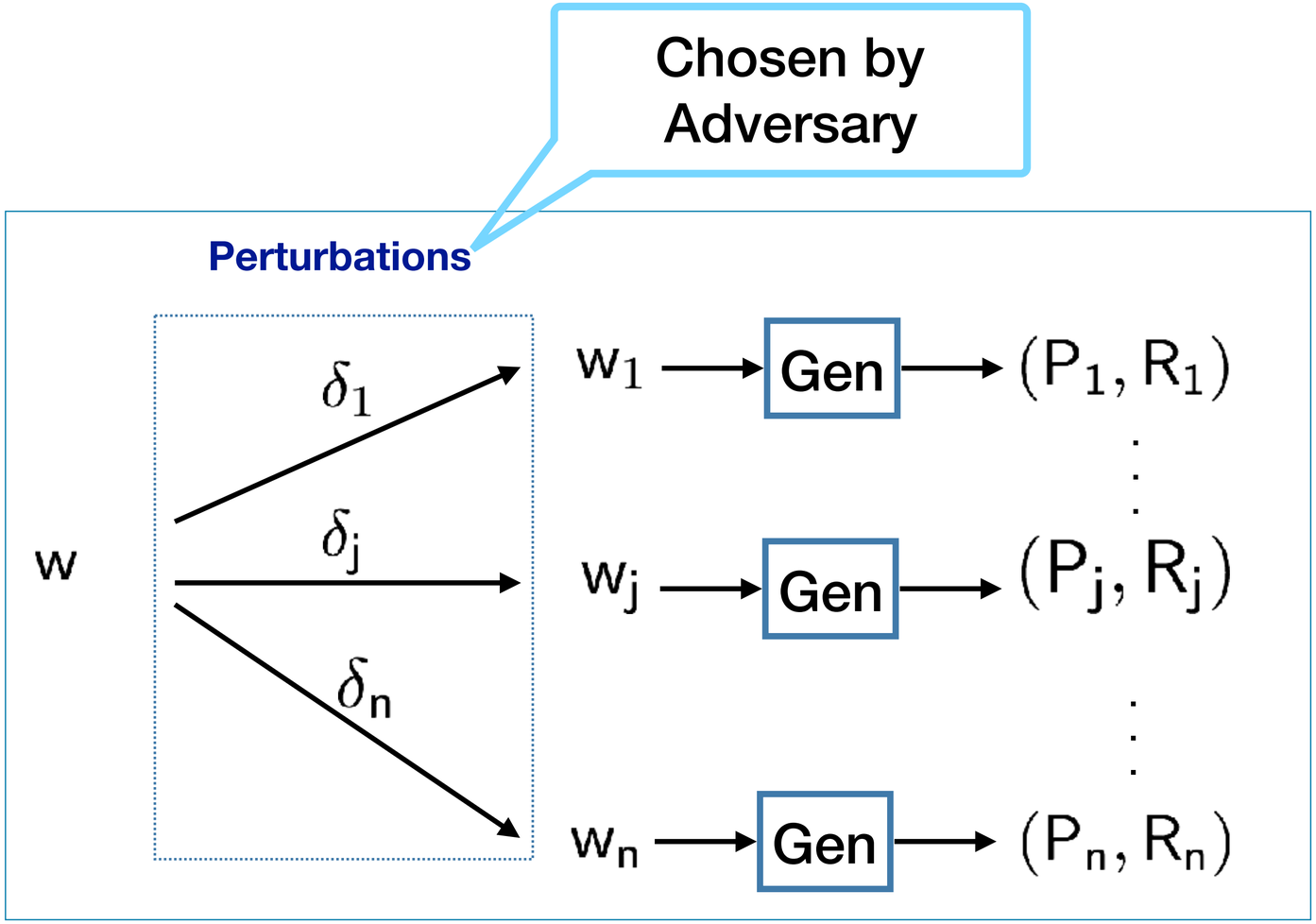
Security：If P is modified, then Rep will output $\perp$.

Reusable Fuzzy Extractor

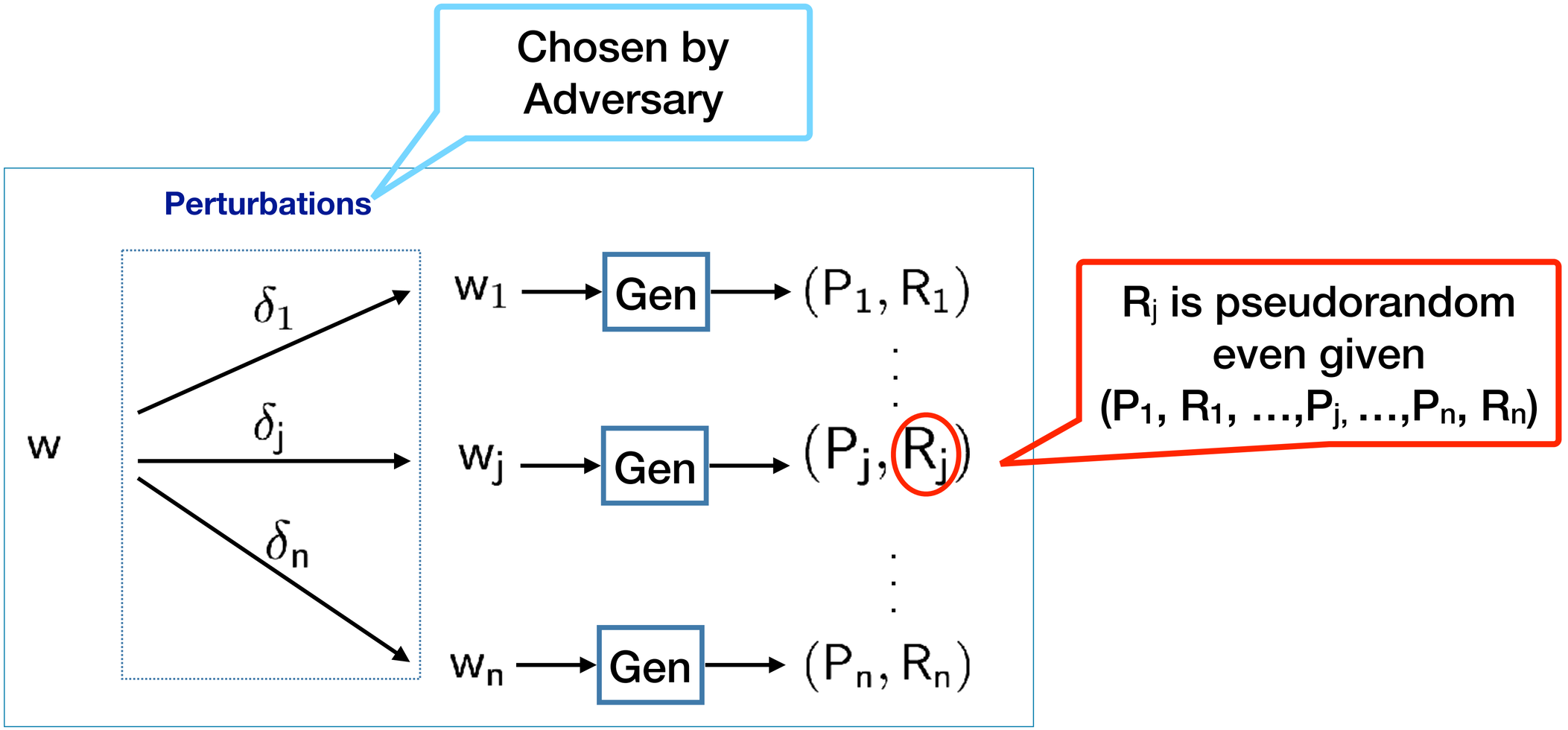


- Biometric is unique and cannot be changed or created.
- The security of multi-extraction from the same noisy source is not guaranteed by fuzzy extractor.

Reusable Fuzzy Extractor



Reusable Fuzzy Extractor



Related Works

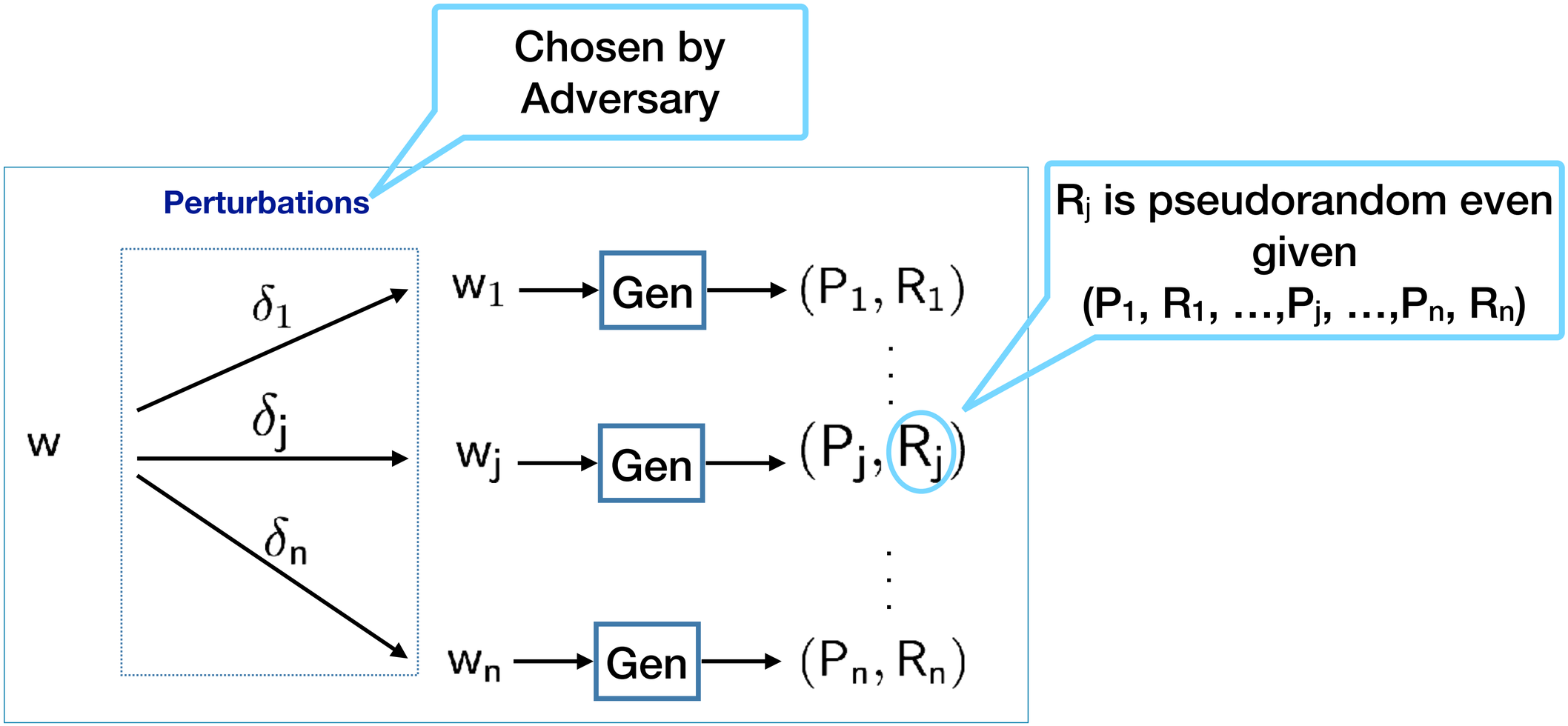
FE schemes	Robustness?	Reusability?
[DRS04], [FMR13]	✗	✗
[Boyen04], [ABCG16], [CFPRS16], [ACEK17], [WL18], [WLH18]	✗	✓
[BDKOS05], [DKRS06], [KR08], [CDFPW08]	✓	✗

No fuzzy extractor considers robustness and reusability simultaneously.

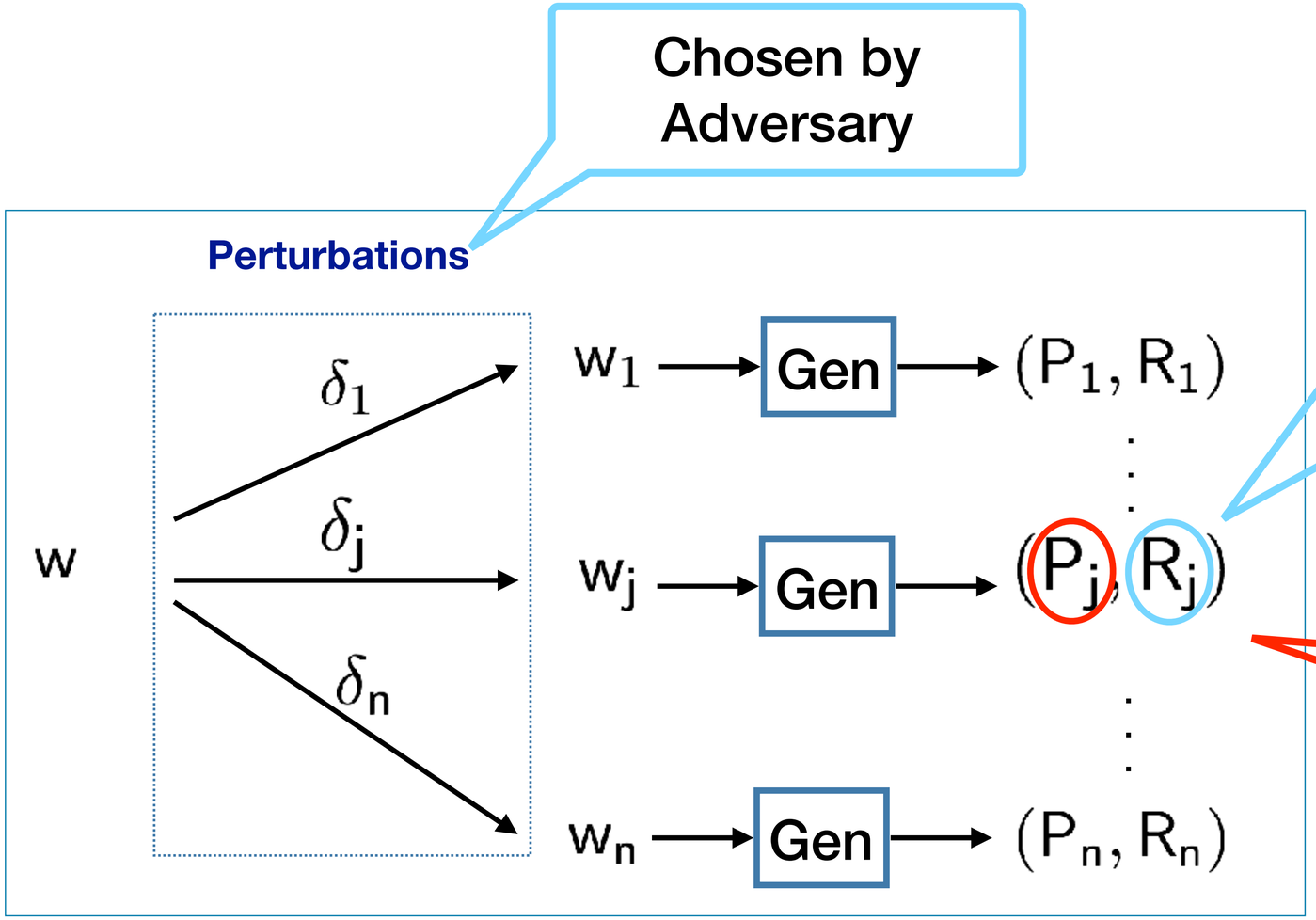
Our Contribution

- We formally defined robustly reusable fuzzy extractor(rrFE).
- We constructed the first rrFE based on standard assumptions.

Robustly Reusable Fuzzy Extractor



Robustly Reusable Fuzzy Extractor



R_j is pseudorandom even given $(P_1, R_1, \ldots, P_j, \ldots, P_n, R_n)$

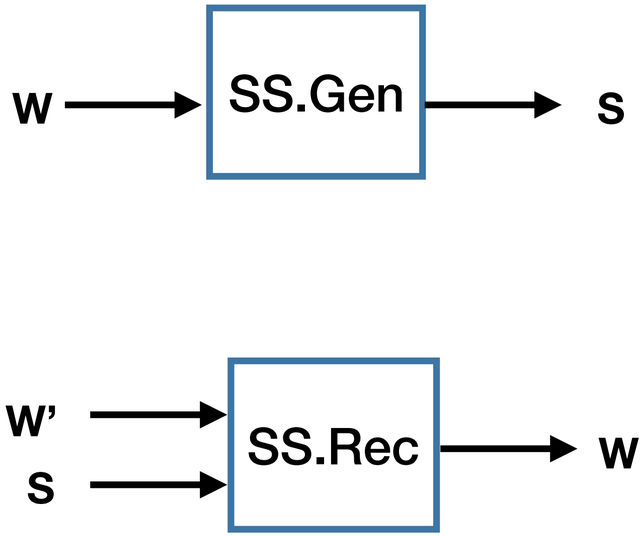
It is hard for adversary to forge P_j', st., Rep does not output bot, even if it gets $(P_1, R_1, \ldots, P_j, R_j, \ldots, P_n, R_n)$.

Building Blocks

- Homomorphic Secure Sketch (SS)
- Homomorphic Extractor (Ext)
- Symmetric Key Encapsulation Mechanism (SKEM)
- Homomorphic Lossy Algebraic Filter (LAF)

Building Block-Secure Sketch

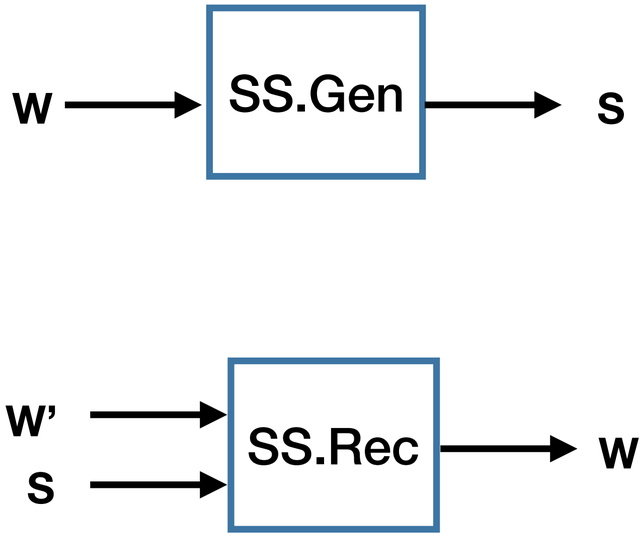
- **SS.Gen(w)**
  - **Input**: a weak random secret w.
  - **Output**: a sketch s.
- **SS.Rec(w', s)**
  - **Input**: a noisy version w' and the sketch s.
  - **Output**: w.



- **Correctness**: For w' close to w, w can be recovered from s.
- **Privacy**: s does not leak too much information of w.

Building Block-Secure Sketch

- **SS.Gen(w)**
  - **Input**: a weak random secret w.
  - **Output**: a sketch s.
- **SS.Rec(w', s)**
  - **Input**: a noisy version w' and the sketch s.
  - **Output**: w.



- **Correctness**: For w' close to w, w can be recovered from s.
- **Privacy**: s does not leak too much information of w.

Homomorphic secure sketch:

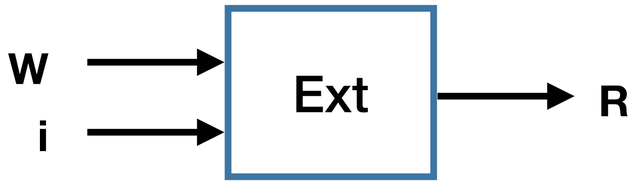
$$SS.Gen(w+w') = SS.Gen(w) + SS.Gen(w').$$

Building Block—Extractor

Extractor

Input: a weak secret w and a uniformly random seed i.

Output: extracted key $R = \mathrm{Ext}(w; i)$.



Security: R is uniformly random, even conditioned on the seed i.

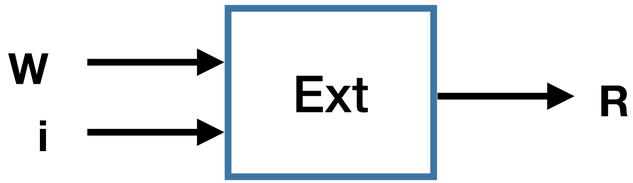
$$(\mathrm{Ext}(W; i), i) \approx (\mathrm{Uniform}, i).$$

Building Block—Extractor

Extractor

Input: a weak secret w and a uniformly random seed i.

Output: extracted key R = Ext(w; i).



Security: R is uniformly random, even conditioned on the seed i.

(Ext(W; i), i) ≈ (Uniform, i).

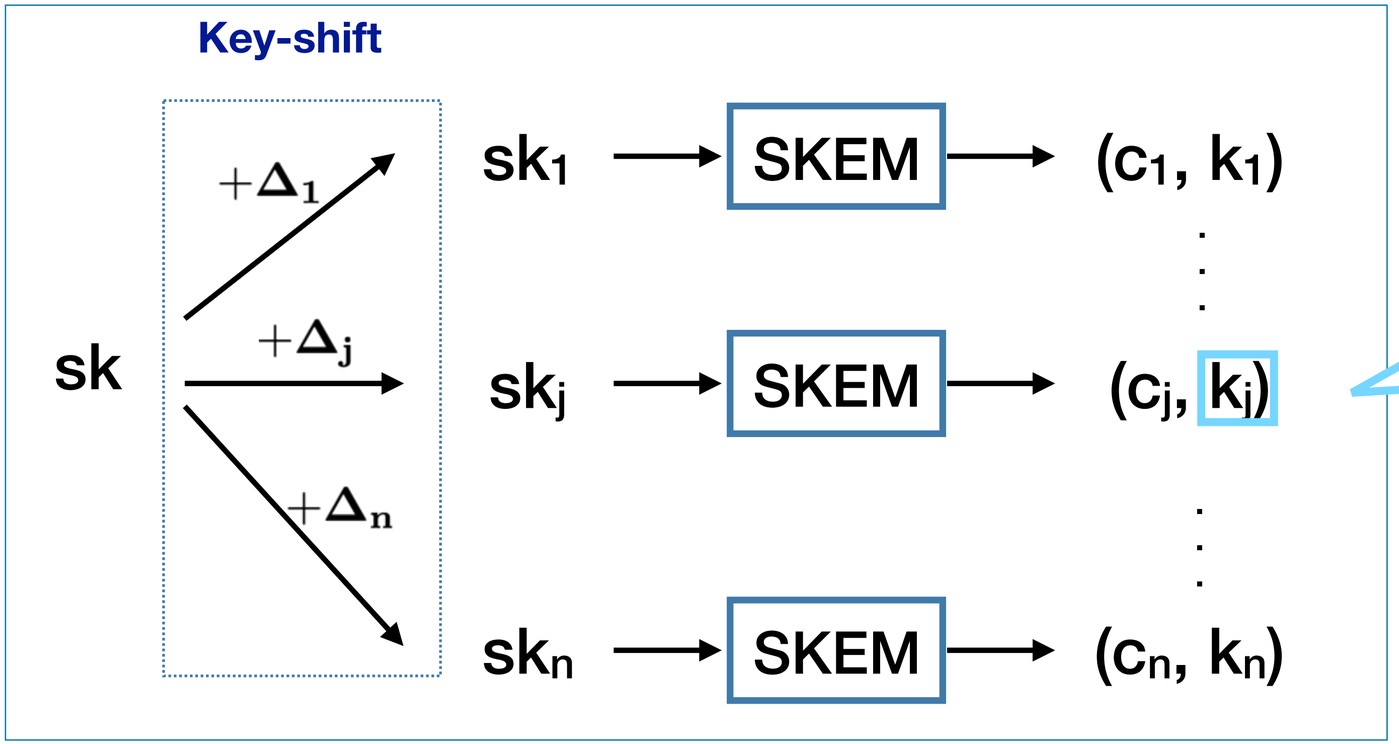
Homomorphic extractor:

Ext(w+w', i) =Ext(w; i) + Ext(w'; i).

Building Block—SKEM

Symmetric Key Encapsulation Mechanism is similar to traditional KEM.

- SKEM.Enc(pp, sk) ⟶ (c, k).
- SKEM.Dec(c, sk)=k.



k_i is pseudorandom even given $(c_1, k_1, \ldots, c_j, \ldots, c_n, k_n)$

Key-Shift Security

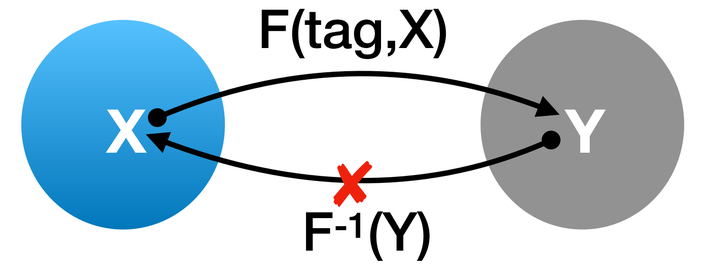
Building Block—LAF

Lossy Algebraic Filter (LAF)

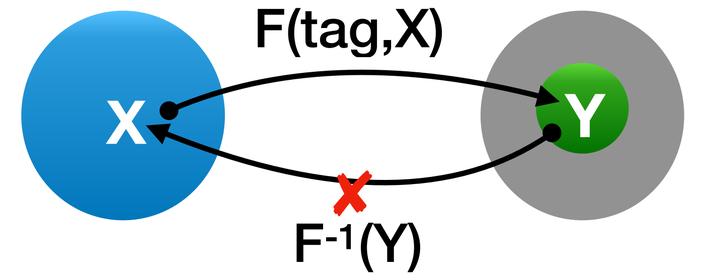
$\mathsf{LAF} : \{\mathsf{F} : \mathcal{X} \to \mathcal{Y}\}$

- $\mathsf{FGen}(1^\lambda) \to (\mathsf{F}_{\mathsf{pk}}, \mathsf{F}_{\mathsf{td}})$.
- $\mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}) \to \mathsf{Y}$.
- $\mathsf{FTag}(\mathsf{F}_{\mathsf{td}}, \mathsf{t}) \to \mathsf{t}'$.

tag=(t, t')



Injective



Lossy

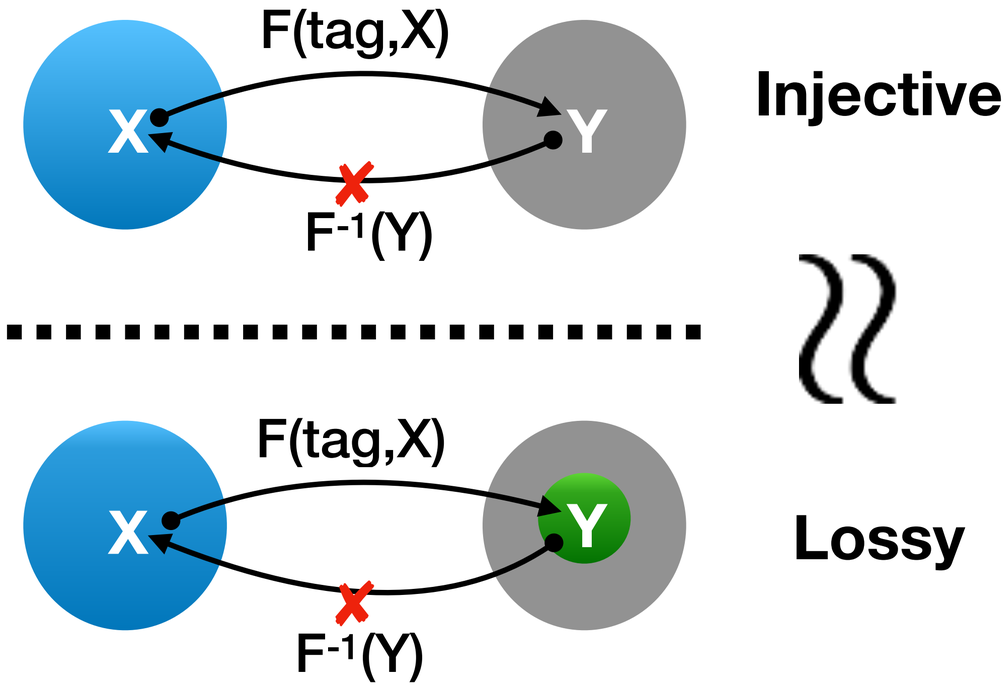
Building Block—LAF

Lossy Algebraic Filter (LAF)

$\mathsf{LAF} : \{\mathsf{F} : \mathcal{X} \to \mathcal{Y}\}$

- $\mathsf{FGen}(1^\lambda) \to (\mathsf{F}_{\mathsf{pk}}, \mathsf{F}_{\mathsf{td}})$.
- $\mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}) \to \mathsf{Y}$.
- $\mathsf{FTag}(\mathsf{F}_{\mathsf{td}}, \mathsf{t}) \to \mathsf{t}'$.

tag=(t, t')



Injective

Lossy

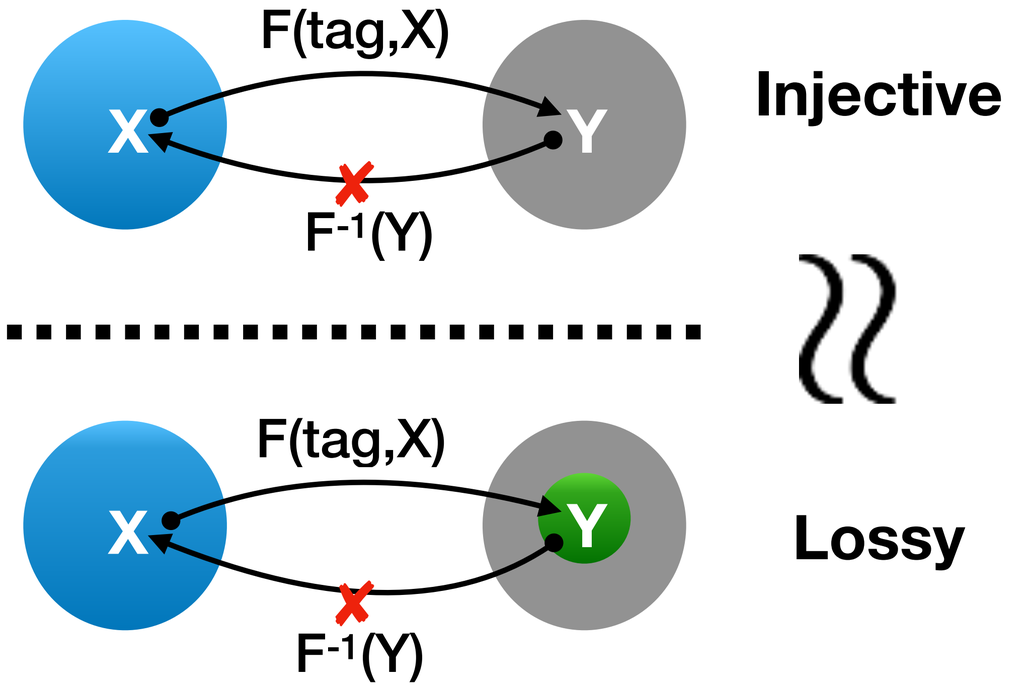
Building Block—LAF

Lossy Algebraic Filter (LAF)

$\mathsf{LAF} : \{\mathsf{F} : \mathcal{X} \to \mathcal{Y}\}$

- $\mathsf{FGen}(1^\lambda) \to (\mathsf{F}_{\mathsf{pk}}, \mathsf{F}_{\mathsf{td}})$.
- $\mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}) \to \mathsf{Y}$.
- $\mathsf{FTag}(\mathsf{F}_{\mathsf{td}}, \mathsf{t}) \to \mathsf{t}'$.

tag=(t, t')



- **Evasiveness**: It is hard to find a non-injective tag without F_{td}.
- **Lossiness**: If the tag is lossy, the function value is only depend on $\sum_{i=1}^{n} u_i X_i$.

Building Block—LAF

Lossy Algebraic Filter (LAF)

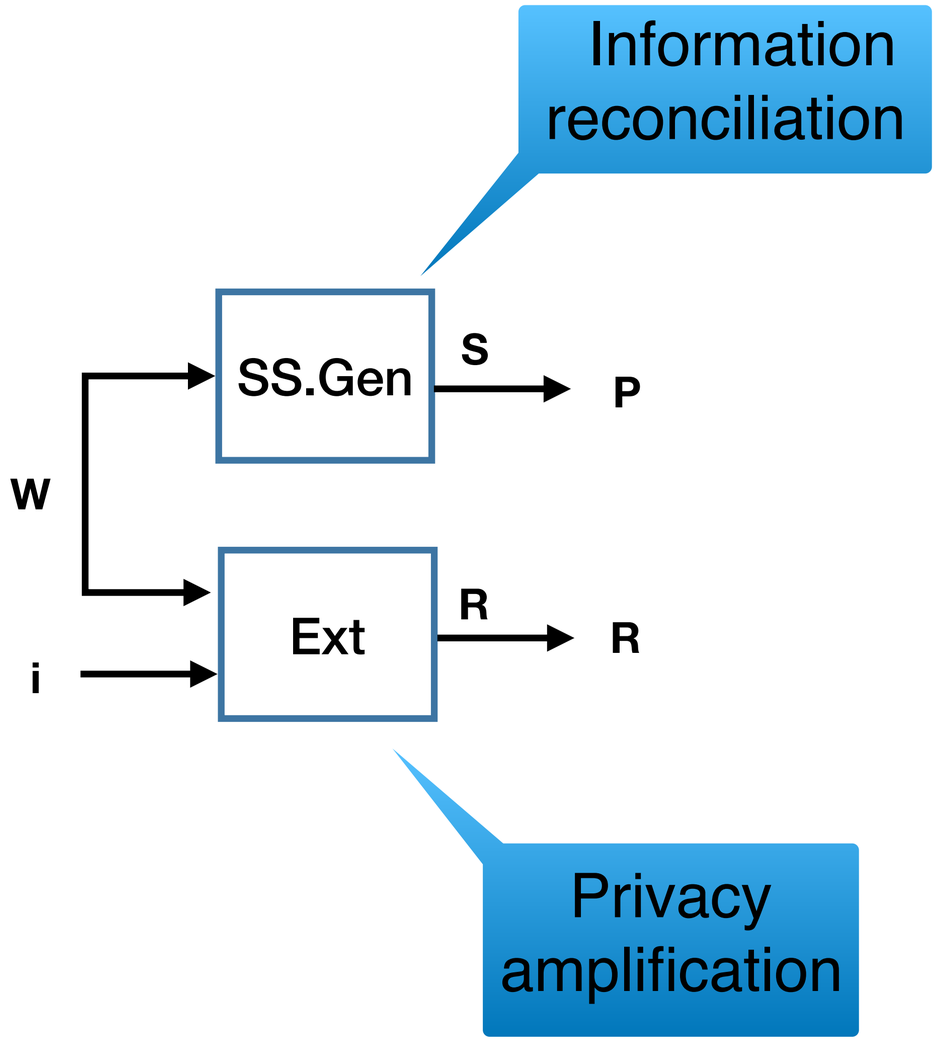
$\mathsf{LAF} : \{\mathsf{F} : \mathcal{X} \to \mathcal{Y}\}$

- $\mathsf{FGen}(1^\lambda) \to (\mathsf{F}_{\mathsf{pk}}, \mathsf{F}_{\mathsf{td}})$.
- $\mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}) \to \mathsf{Y}$.
- $\mathsf{FTag}(\mathsf{F}_{\mathsf{td}}, \mathsf{t}) \to \mathsf{t}'$.

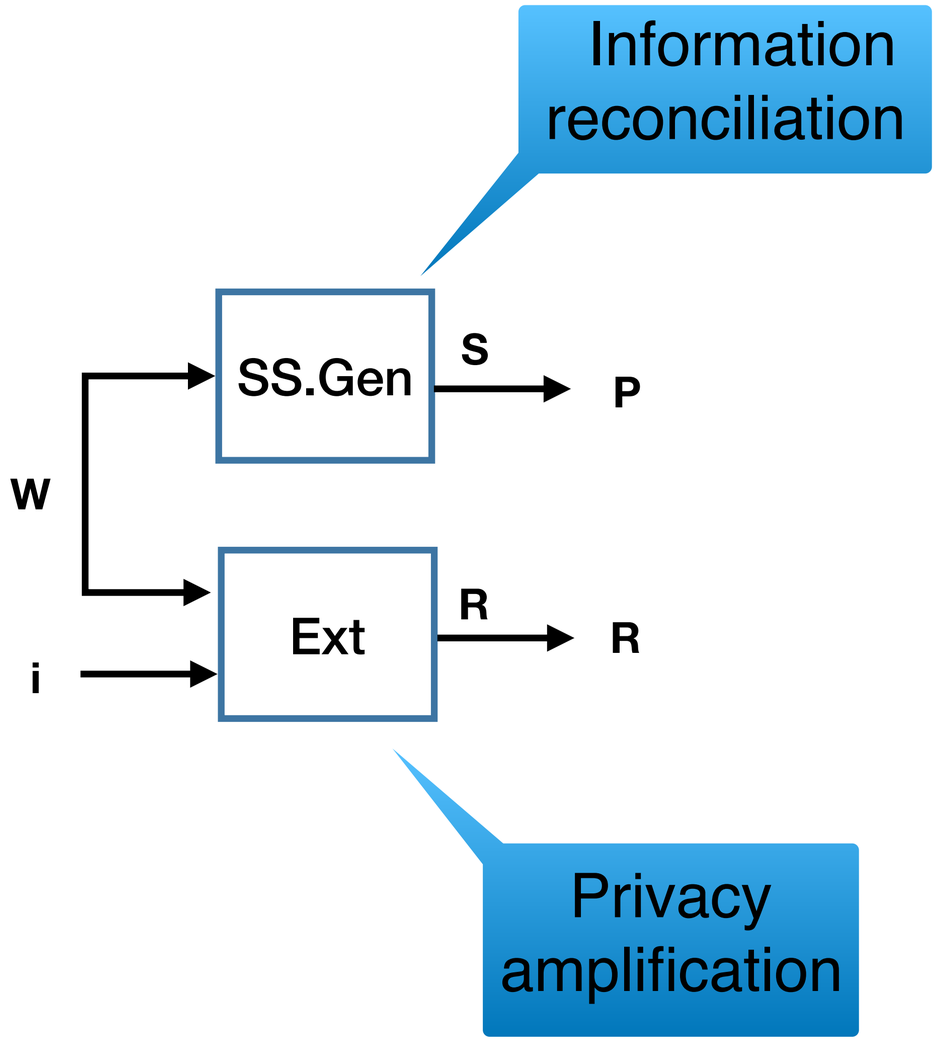
Homomorphic LAF:

$$\mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}_1 + \mathsf{X}_2) = \mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}_1) + \mathsf{FEval}(\mathsf{F}_{\mathsf{pk}}, \mathsf{tag}, \mathsf{X}_2)$$

Sketch-and-Extract Paradigm

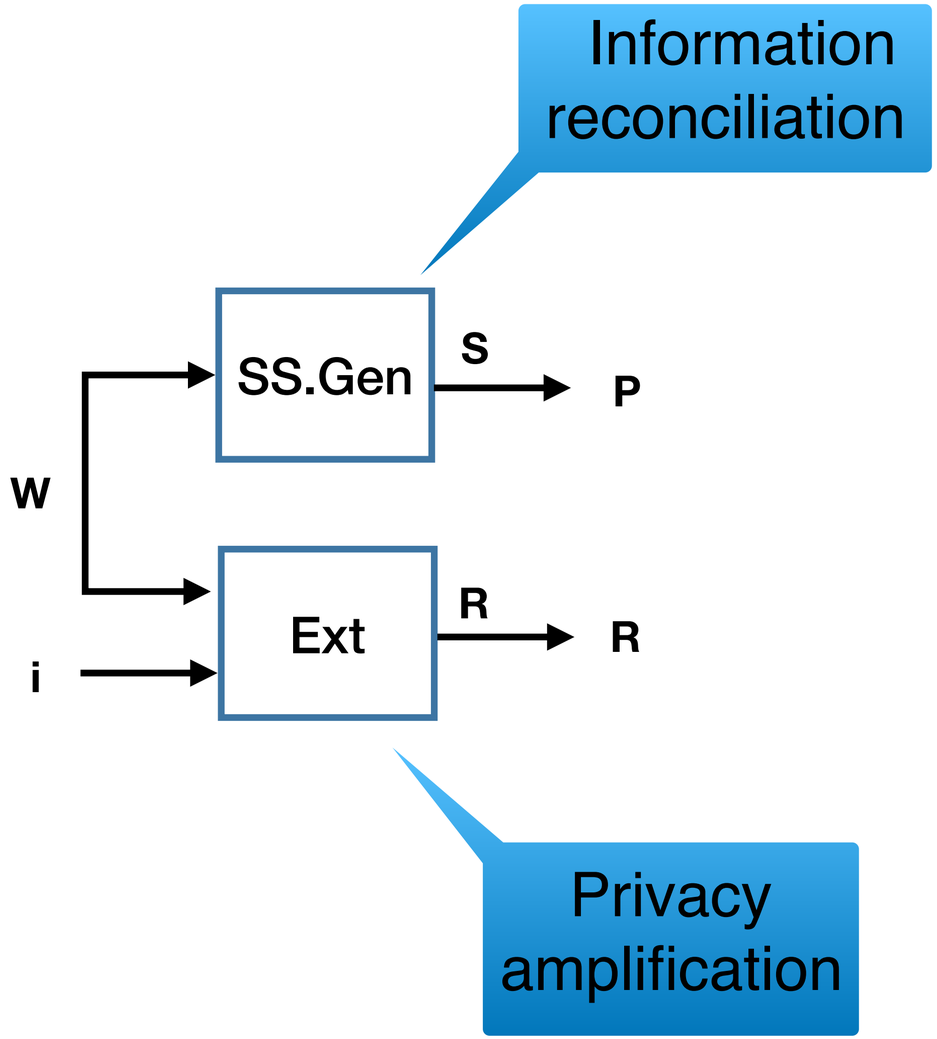


Sketch-and-Extract Paradigm



The “sketch-and-extract” construction is a fuzzy extractor [DRS04].

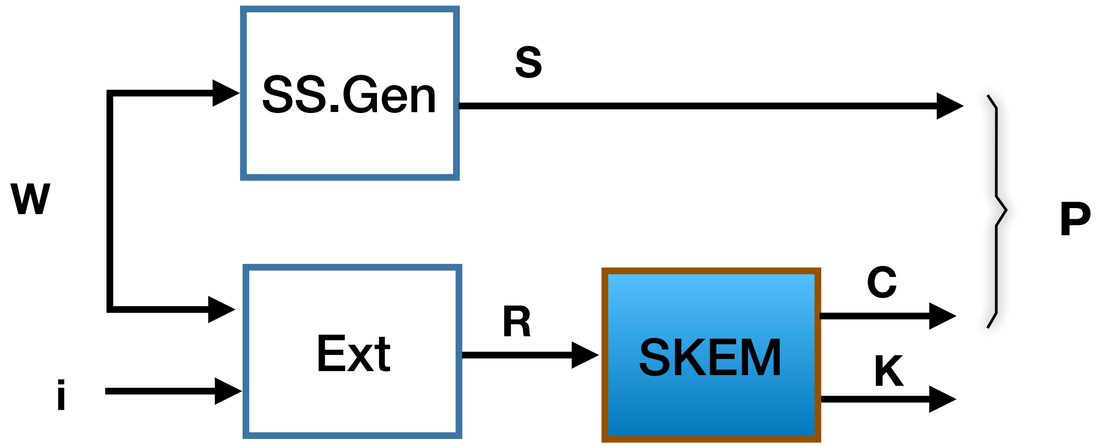
Sketch-and-Extract Paradigm



The “sketch-and-extract” construction is a fuzzy extractor [DRS04].

Not reusable:
Same w, same R.
Not robust:
No authentication

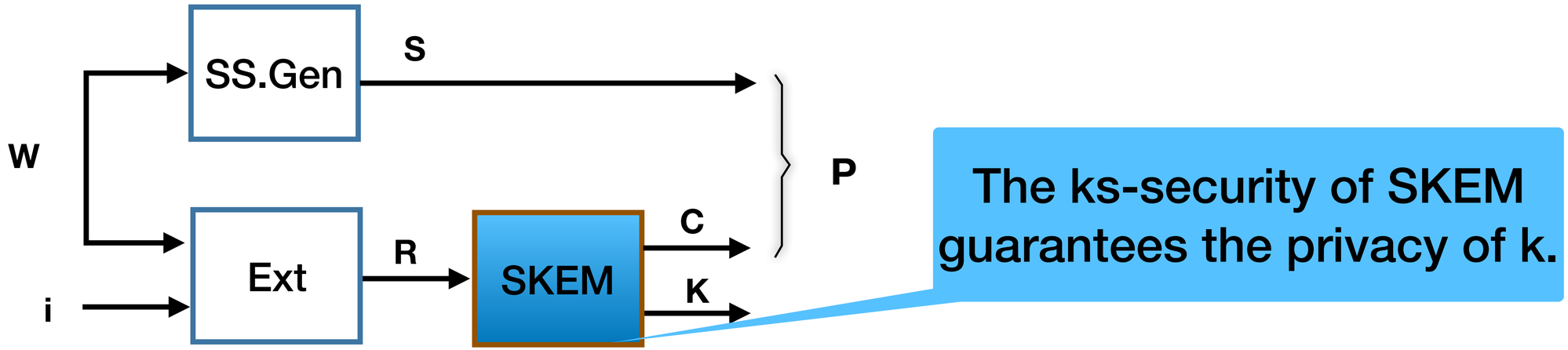
How to Achieve Reusability



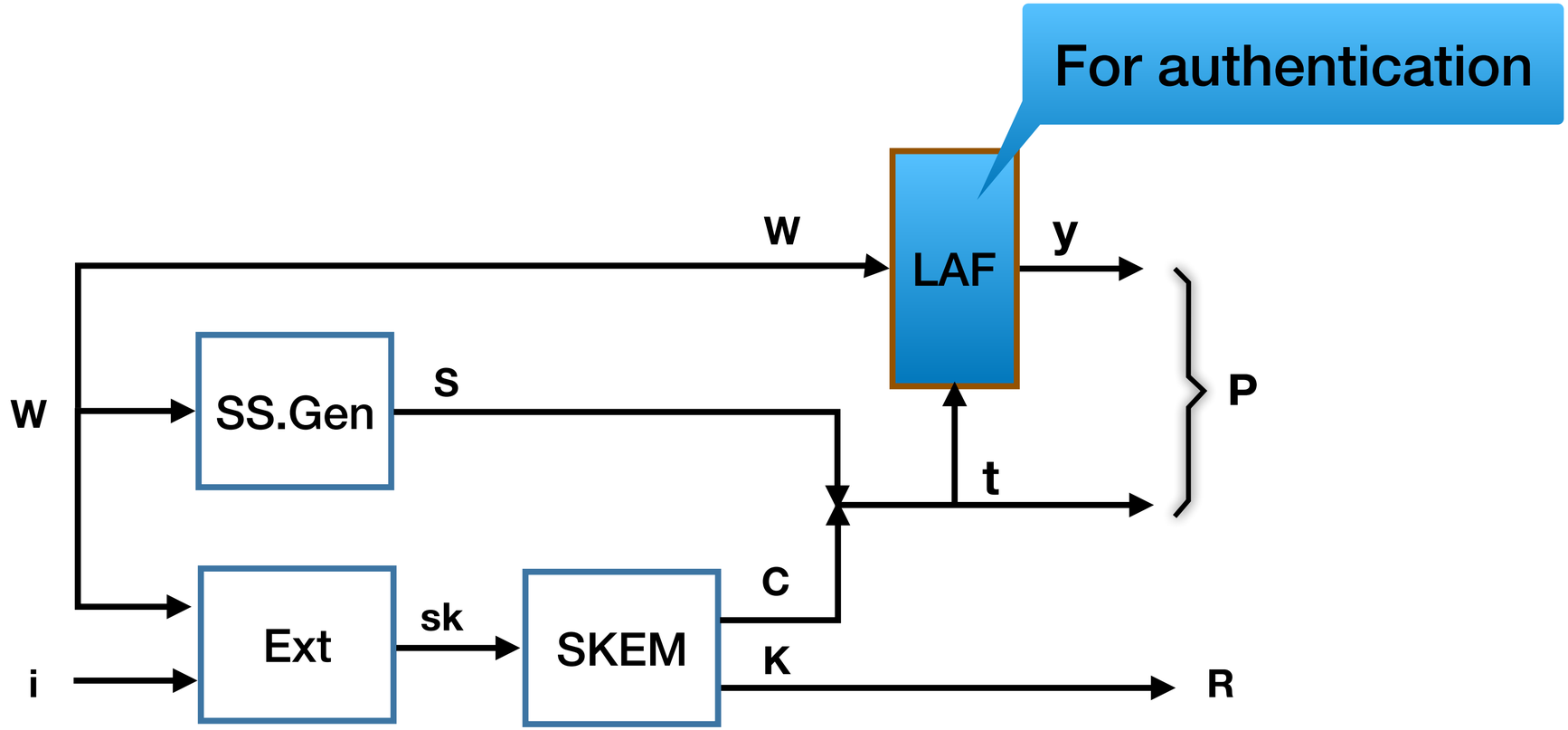
How to Achieve Reusability

Homomorphic properties:

- $s_j = SS.Gen(w + \delta_j) = SS.Gen(w) + SS.Gen(\delta_j)$.
- $R_j = Ext(w + \delta_j, i) = Ext(w, i) + Ext(\delta_j, i)$.



How to Achieve Robustness



Seed i of Ext, Fpk of LAF and pp of SKEM are common reference string, which can be stored publicly, but can not be modified.

Reusability & Robustness

- $s_j = \mathsf{SS.Gen}(w + \delta_j) = \mathsf{SS.Gen}(w) + \mathsf{SS.Gen}(\delta_j)$.
- $sk_j = \mathsf{Ext}(w + \delta_j, i) = \mathsf{Ext}(w, i) + \mathsf{Ext}(\delta_j, i)$.
- $y_j = \mathsf{FEval}(F_{pk}, tag, w + \delta_j) = \mathsf{FEval}(F_{pk}, tag, w) + \mathsf{FEval}(F_{pk}, tag, \delta_j)$.

All tags are changed into lossy tags.

Reusability & Robustness

- $s_j = SS.Gen(w + \delta_j) = SS.Gen(w) + SS.Gen(\delta_j)$.
- $sk_j = Ext(w + \delta_j, i) = Ext(w, i) + Ext(\delta_j, i)$.
- $y_j = FEval(F_{pk}, tag, w + \delta_j) = FEval(F_{pk}, tag, w) + FEval(F_{pk}, tag, \delta_j)$.

All tags are changed into lossy tags.

Enough entropy is left for Ext to extract a key and for LAF to authenticate.

Instantiation

- Homomorphic Ext and SS have information theoretical instantiations.
- Homomorphic LAF can be constructed by the DLIN assumption.
- Key-shift secure SKEM can be constructed by the DDH assumption.

Our rrFE is based on standard assumptions.

Summary

- Our contribution
  - We constructed the first robustly reusable fuzzy extractor from standard assumption.
- Open problem
  - Robustly reusable FE for arbitrary correlated inputs.

Thank you !