how to do it right...

A Leakage-Resilient Mode of Operation for Block-Ciphers

Krzysztof Pietrzak (CWI Amsterdam)

crypto'08 rump session

- How to use a block-cipher F to get a stream-cipher S^F which is secure against side-channel attacks???

- How to use a block-cipher F to get a stream-cipher S^F which is secure against side-channel attacks???
- Not like in Keeloq.

- How to use a block-cipher F to get a stream-cipher S^F which is secure against side-channel attacks???
- Not like in Keeloq.
- Want security against all side-channels: leakage function adaptively and adversarially chosen.

- Will use definition of leakage-resilient stream-cipher from [Dziembowski,P FOCS’08] (there we use PRG & extractor, now just a PRF)

- Will use definition of leakage-resilient stream-cipher from [Dziembowski,P FOCS’08] (there we use PRG & extractor, now just a PRF)
- S^{F} outputs $M_1, M_2, \ldots$ ($M_i \in \{0,1\}^n$).

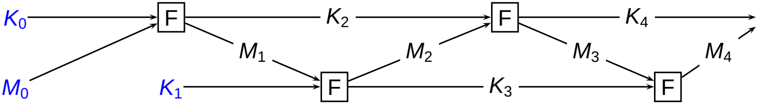
- Will use definition of leakage-resilient stream-cipher from [Dziembowski,P FOCS'08] (there we use PRG & extractor, now just a PRF)
- S^{F} outputs $M_1, M_2, \ldots$ ($M_i \in \{0,1\}^n$).
- Before M_i is output, adversary chooses leakage function f and gets output of f applied to the state of S^{F} (not quite).

- Will use definition of leakage-resilient stream-cipher from [Dziembowski,P FOCS'08] (there we use PRG & extractor, now just a PRF)
- S^{F} outputs $M_1, M_2, \ldots$ ($M_i \in \{0,1\}^n$).
- Before M_i is output, adversary chooses leakage function f and gets output of f applied to the state of S^{F} (not quite).
- Bounded Leakage: Range of f is bounded to λ bits.

- Will use definition of leakage-resilient stream-cipher from [Dziembowski,P FOCS'08] (there we use PRG & extractor, now just a PRF)
- S^{F} outputs $M_1, M_2, \ldots$ ($M_i \in \{0,1\}^n$).
- Before M_i is output, adversary chooses leakage function f and gets output of f applied to the state of S^{F} (not quite).
- Bounded Leakage: Range of f is bounded to λ bits.
- Only computation leaks information: f gets as input only the part of the state that is actually accessed to compute M_i.

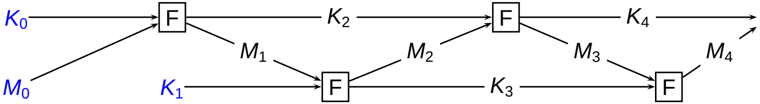
Leakage Resilient Stream cipher.

- PRF F : $\{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^{\kappa+n}$ e.g. $\mathsf{F}(K,X) = AES(K,0\|X)\|AES(K,1\|X)$
- Secret key is K_0, K_1, M_0, output is $M_0, M_1, \ldots$
- i'th round: $(K_{i+2}, M_{i+1}) = \mathsf{F}(K_i, M_i)$.



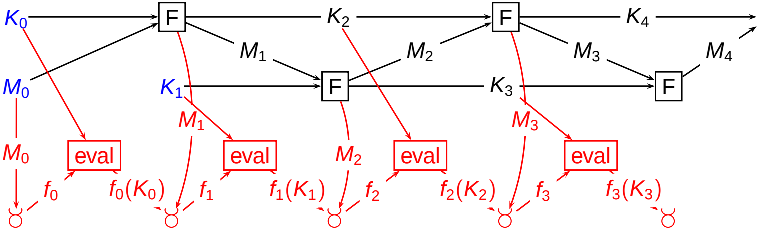
Leakage Resilient Stream cipher.

- PRF F : $\{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^{\kappa+n}$
  e.g. $F(K,X) = AES(K,0\|X)\|AES(K,1\|X)$
- Secret key is K_0, K_1, M_0, output is $M_0, M_1, \ldots$
- i'th round: $(K_{i+2}, M_{i+1}) = F(K_i, M_i)$.



Leakage Resilient Stream cipher.

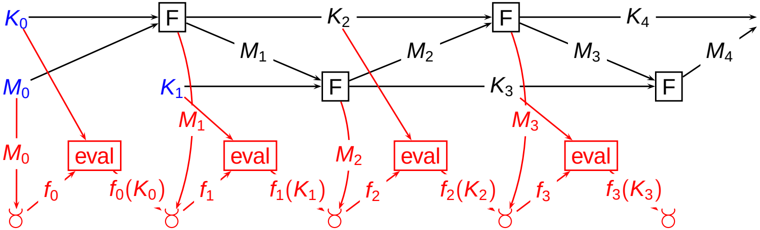
- PRF $\mathsf{F} : \{0,1\}^{\kappa} \times \{0,1\}^{n} \rightarrow \{0,1\}^{\kappa+n}$
  e.g. $\mathsf{F}(K, X) = AES(K, 0\|X)\|AES(K, 1\|X)$
- Secret key is K_0, K_1, M_0, output is $M_0, M_1, \ldots$
- i'th round: $(K_{i+2}, M_{i+1}) = \mathsf{F}(K_i, M_i)$.



- Round i: attacker ♉ chooses f_{i-1} and gets leakage $f_{i-1}(K_{i-1})$ and output M_i.

Leakage Resilient Stream cipher.

- PRF $\mathsf{F} : \{0,1\}^{\kappa} \times \{0,1\}^{n} \rightarrow \{0,1\}^{\kappa+n}$
  e.g. $\mathsf{F}(K,X) = AES(K,0\|X)\|AES(K,1\|X)$
- Secret key is K_0, K_1, M_0, output is $M_0, M_1, \ldots$
- i'th round: $(K_{i+2}, M_{i+1}) = \mathsf{F}(K_i, M_i)$.



- Round i: attacker ♉ chooses f_{i-1} and gets leakage $f_{i-1}(K_{i-1})$ and output M_i.
- Security: M_ℓ is pseudorandom given $M_1, \ldots, M_{\ell-1}$ and $f_0(K_0), \ldots, f_{\ell-1}(K_{\ell-1})$.

Tools

- (from [DP'08]) For any PRG $\mathsf{G} : \{0,1\}^m \to \{0,1\}^n$ and any function $f : \{0,1\}^m \to \{0,1\}^\lambda$: $\mathsf{G}(S)$ has high HILL pseudoentropy even given $f(S)$.
- (new) Any weak PRF is seed compressible.