TAMPER DETECTION AND NON-MALLEABLE CODES

Daniel Wichs (Northeastern U)

Protecting Data Against "Tampering"

- Question: How can we protect data against tampering by an adversary?

- Variants of this question studied in cryptography, information theory and coding theory.
  - What kind of tampering are we considering?
  - What protection/guarantees do we want to achieve?
  - Can we use secret keys or randomness ?
- Tools: Signatures, MACs, Hash Functions, Error-correcting codes, Error-detecting codes.

- New variants: tamper-detection codes, non-malleable codes.

Motivation: Physical Attacks

- Goal: store secret data on a device
- Adversary cannot read the data on the device directly, but can:
  - interact with the device via interface
  - tamper with the data on the device.

Motivating Example (Signature)

- If a single bit of the signing key is flipped, can use the resulting signature to factor the RSA modulus. [BDL97]

m →  → $\mathrm{Sign}_{sk}(m)$

sk

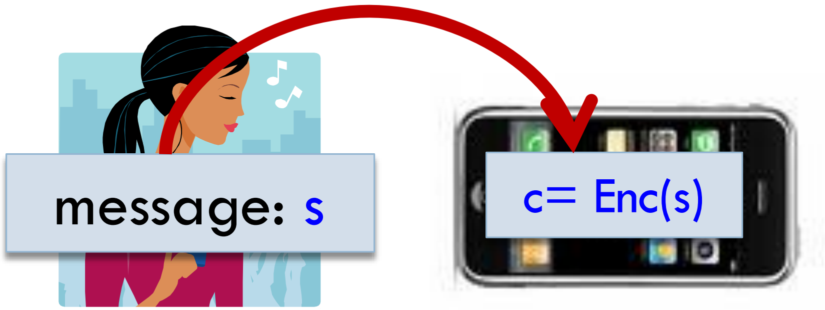
Coding against Tampering

- Solution Idea: encode the data on the device to protect it against tampering.
  - Each execution first decodes the underlying data.

- Example: Use an error-correcting code to protect against attacks that modify a few bits.

- What kind of tampering can we protect against?
- What kind of codes do we need?

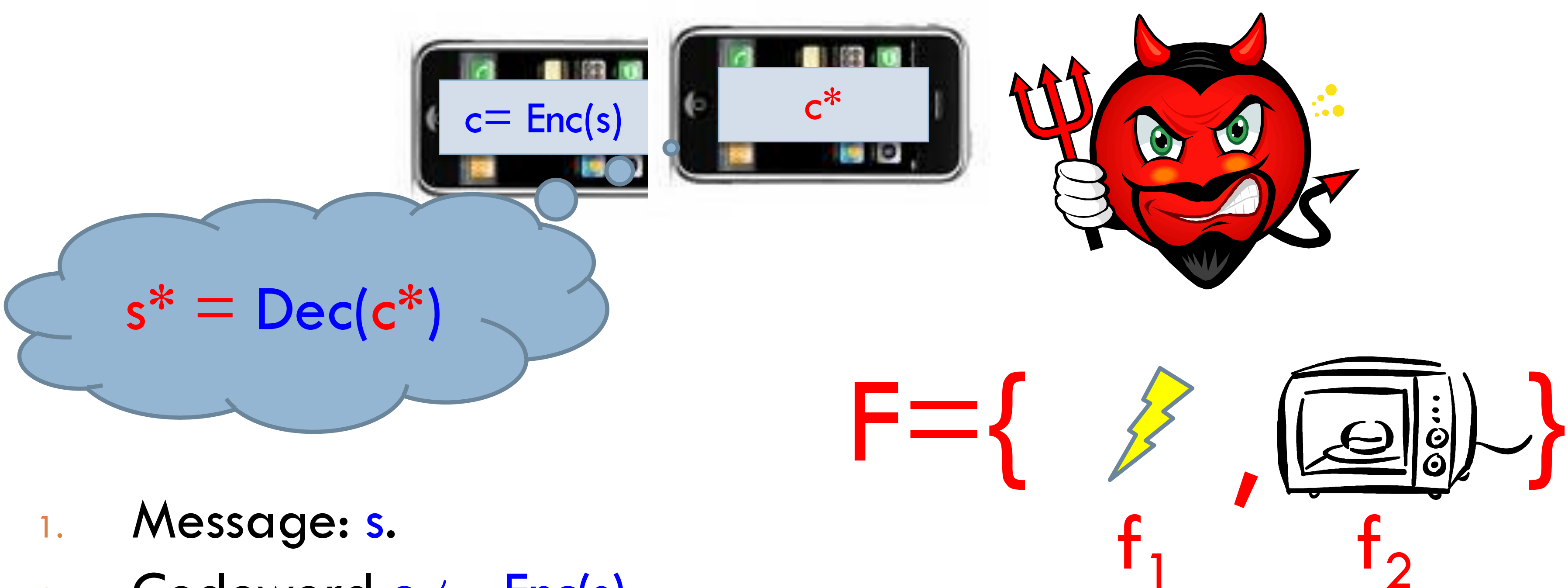
The "Tampering Experiment"



- Coding scheme (Enc, Dec) s.t.
  - Enc : $\{0,1\}^k \rightarrow \{0,1\}^n$
    can be randomized
  - Dec(Enc(s)) = s
    (with probability 1)

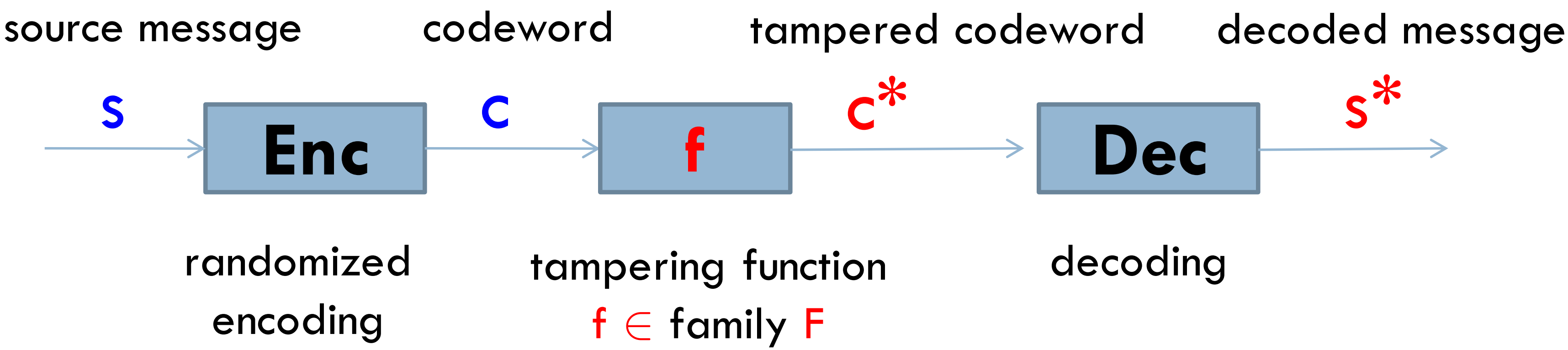
1. Message: s.
2. Codeword $c \leftarrow$ Enc(s).

The “Tampering Experiment”



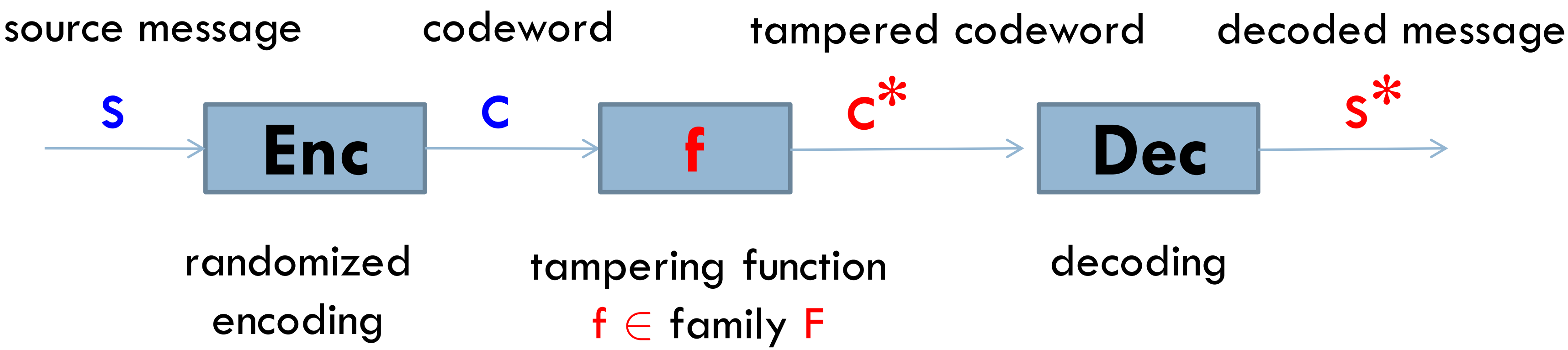
1. Message: s.
2. Codeword $c \leftarrow \mathrm{Enc}(s)$.
3. Tampered codeword $c^* = f(c)$.
   - $f \in F$ adversarial but independent of randomness of c.
4. Decoded message: $s^* = \mathrm{Dec}(c^*)$.

The “Tampering Experiment”



- Differences from “standard” coding problems:
  - No notion of *distance* between original and tampered codeword. Focus on the family of functions being applied.
  - Tampering is “worst-case”, but choice of function f does not depend on randomness of encoding.

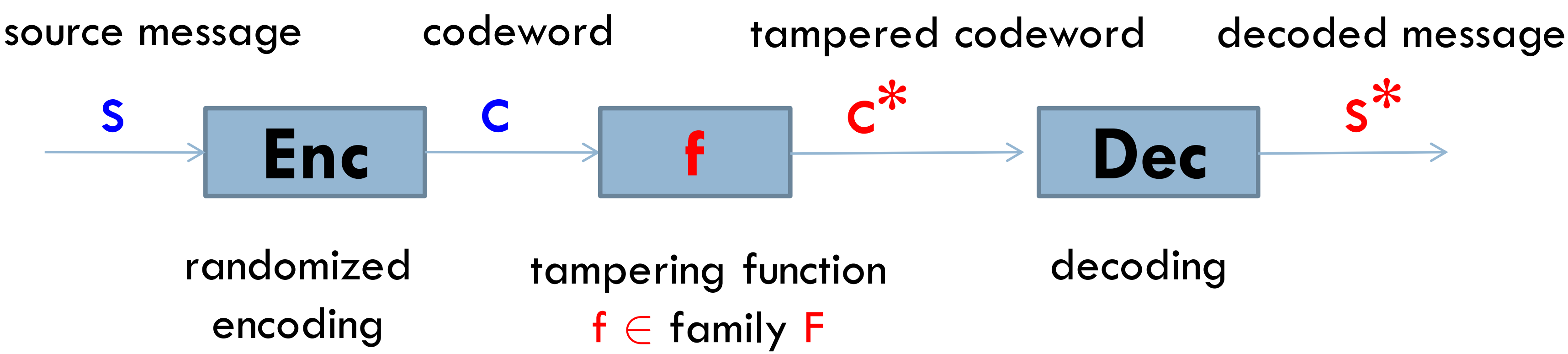
The "Tampering Experiment"



Goal:

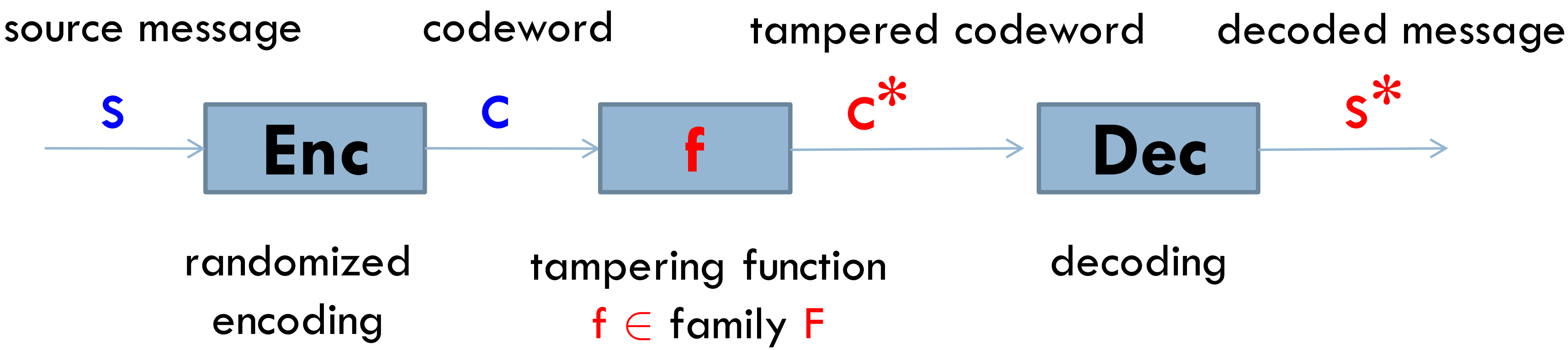
For "interesting" families F, design coding scheme (Enc, Dec) which provides "meaningful guarantees" about the outcome of the tampering experiment.

Correction



- <u>Tamper-Correction:</u> require that $s^* = s$
- Error-Correcting Codes for Hamming Distance: The family
  $$F = \{f \text{ s.t. } \forall x \ \text{dist}(x, f(x)) < d\}$$
- Too limited for us! Must preserve some relationship between original and tampered codeword.
  - E.g., cannot protect against overwriting with random value.

Tamper Detection



- <u>Tamper-Detection:</u> If tampering occurs, then we require that $s^* = \perp$ (error) with overwhelming probability.

- **Definition:** An (F, ε)-*Tamper Detection Code* guarantees:

$$\forall\, s, f \in F :\ \Pr[\, Dec(\, f(\, Enc(s)\,)\,) \neq \perp \,] \leq \varepsilon$$

Tamper Detection

An (F, ε)-*Tamper Detection Code* guarantees:

$$\forall\, s, f \in F \;:\; \Pr[\, Dec(\, f(\, Enc(s)\,)\,) \neq \perp \,] \leq \varepsilon$$

- Error-Correcting Codes provide tamper detection for the family $F = \{f \;\; \text{s.t.} \;\; \forall\, x \;\; 0 < dist(x, f(x)) < d\, \}$

Tamper Detection: AMD Codes

An (F, ε)-*Tamper Detection* Code guarantees:

$$\forall\, s, f \in F \;:\; \Pr[\,\mathrm{Dec}(\, f(\,\mathrm{Enc}(s)\,)\,) \neq \perp\,] \leq \varepsilon$$

- Algebraic Manipulation Detection (AMD) Codes [CDFPW08] :
  Tamper detection for $F = \{\, f_e(x) = x + e : e \neq 0 \,\}$

- Intuition: Can add any error e you want, but must choose it before you see the codeword.

- Encoding is necessarily randomized. Choice of $f_e \in F$ must be independent of randomness.

Tamper Detection: AMD Codes

An (F, ε)-*Tamper Detection* Code guarantees:

$$\forall\, s\,,\, f \in F \;:\; \Pr[\,\mathrm{Dec}(\,f(\,\mathrm{Enc}(s)\,)\,) \neq \perp\,] \leq \varepsilon$$

- Algebraic Manipulation Detection (AMD) Codes [CDFPW08] :

  Tamper detection for $F = \{\, f_e(x) = x + e : e \neq 0 \,\}$

- Construction: $\mathrm{Enc}(s) = (s, r, sr + r^3)$ operations in $\mathbb{F}_{2^k}$.
  - Proof Idea: $\mathrm{Enc}(s) + e$ is valid iff $p(r) = 0$ where p is a non-zero poly of $\deg(p) \leq 2$.

- Construction Generalizes to get a rate 1 code:

  Message size k, codeword size $n = k + O(\log k + \log 1/\varepsilon)$

Tamper Detection: AMD Codes

An $(\mathcal{F}, \varepsilon)$-*Tamper Detection Code* guarantees:

$$\forall\, s\,,\, f \in \mathcal{F} \;:\; \Pr[\, \mathrm{Dec}(\, f(\, \mathrm{Enc}(s)\,)\,) \neq \perp \,] \leq \varepsilon$$

- Algebraic Manipulation Detection (AMD) Codes [CDFPW08] :
  Tamper detection for $\mathcal{F} = \{\, f_e(x) = x + e \,:\; e \neq 0 \,\}$

- Many applications of AMD codes:
  - Secret Sharing and Fuzzy Extractors [CDFPW08]
  - Error-Correcting Codes for "Simple" Channels [GS10]
  - Multiparty Computation [GIPST14]
  - Related-Key Attack Security
  - …

Tamper Detection: Beyond AMD?

An (F, ε)-*Tamper Detection* Code guarantees:

$$\forall\, s\,,\, f \in F\ :\ \Pr[\, Dec(\, f(\, Enc(s)\,)\,)\ \neq \perp\,] \leq \varepsilon$$

Question: Can we go beyond AMD codes?

- What function families F allow for tamper-detection codes?

- Can't allow functions that are (close to) "identity".
- Can't allow functions that are (close to) "constant".
- Can't allow functions that are "too complex":
  - e.g., $f(x) = Enc(\, Dec(x) + 1\,)$

Tamper Detection: General Result

Theorem [Jafargholi-W15]:

For any function family F over n-bit codewords, there is an (F, ε)-*TDC* as long as $|F| < 2^{2^{\alpha n}}$ for $\alpha < 1$ and each $f \in F$ has *few fixed points* and *high entropy*.

- Few fixed-points: $\Pr_x[\ f(x) = x]$ is small.
- High entropy: $\forall$ c: $\Pr_x[\ f(x) = c]$ is small.

Rate of code is $\approx 1 - \alpha$

Tamper Detection: General Result

Theorem [Jafargholi-W15]:

For any function family F over n-bit codewords, there is an (F, ε)-*TDC* as long as $|F| < 2^{2^{\alpha n}}$ for $\alpha < 1$ and each $f \in F$ has *few fixed points* and *high entropy*.

- Proof is via probabilistic method argument - construction is inherently inefficient.
- Can be made efficient for $|F| = 2^{\mathrm{poly}(n)}$.
- Examples:
  - F = { Polynomials p(x) of "low" degree}
  - F = { Affine functions Ax + b over "large" field}

Tamper Detection: Construction

- First, focus on *weak* TDC (random-message security):

  $\forall f \in F : \Pr_s [\text{Dec}(f(\text{Enc}(s))) \neq \perp] \leq \varepsilon$

- Family of codes indexed by function $h : \{0,1\}^k \rightarrow \{0,1\}^v$

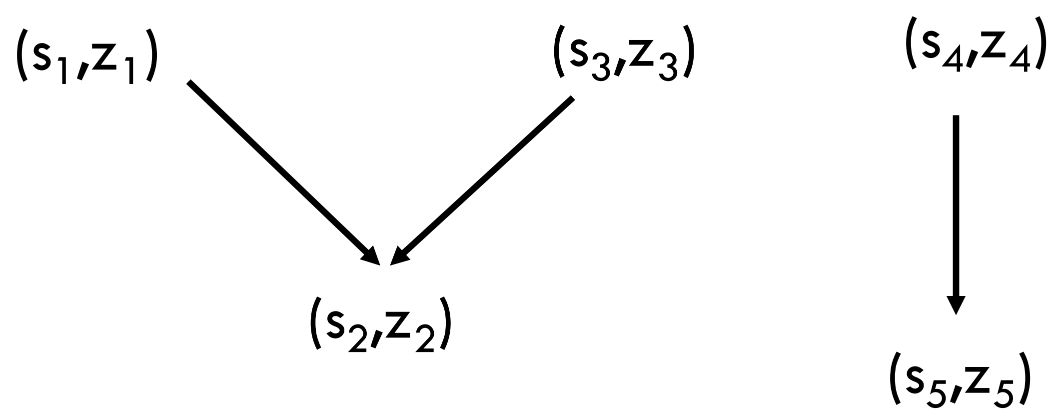
  $\text{Enc}_h(s) = (s, h(s))$ and $\text{Dec}_h(s,z) = \{ s \text{ if } z = h(s) \text{ else } \perp \}$
  - Output size v is $\log(1/\varepsilon) + O(1)$ bits.

- For any family F with given restrictions, a random code $(\text{Enc}_h, \text{Dec}_h)$ is a wTDC with overwhelming probability.
  - Can choose h from a t-wise indep function family where $t = \log|F|$.

Tamper Detection: Analysis

Construction: $\mathrm{Enc}_h(s) = (s, h(s))$, $\mathrm{Dec}_h(s,z) = \{ s \text{ if } z = h(s) \text{ else } \perp \}$

Represent tampering function f as a graph:



Bad edge:
$z = h(s)$
for both
end points

- When is (Enc_h , Dec_h) a bad code? Too many bad edges!
- Unfortunately, "badness" is not independent.
- Can edge-color this graph with few colors (low in-degree). Within each color, "badness" is independent.

Tamper Detection: Construction

- Can go from weak to strong tamper detection via *leakage resilient (LR) codes.*

- **Definition [DDV10]**: A code (LREnc, LRDec) is an (F, ℓ, ε)-LR code if $\forall$ s, $\forall$ f $\in$ F where $f : \{0,1\}^n \rightarrow \{0,1\}^\ell$ we have:

$$f(\mathrm{LREnc}(s)) \approx_\varepsilon f(\mathrm{Uniform})$$

- Construction $\mathrm{LREnc}_h(s) = (r, h(r) + s)$
  - Size of randomness r is $\max\{\ell, \log\log|F|\} + O(\log 1/\varepsilon)$.
  - Can use t-wise indep function h where $t = O(|\log F|)$.

Tamper Detection: Construction

- Can go from weak to strong tamper detection via *leakage resilient (LR) codes.*

- **Definition [DDV10]**: A code (LREnc, LRDec) is an (F, ℓ, ε)-LR code if $\forall$ s, $\forall$ f $\in$ F where f : $\{0,1\}^n \rightarrow \{0,1\}^\ell$ we have:

  $$f(\text{LREnc}(s)) \approx_\varepsilon f(\text{Uniform})$$

- Strong Tamper-Detection: Enc(s) = wtdEnc(LREnc(s))
- Tamper f $\Rightarrow$ Leak f'(c) = {1 if wtdDec(c) $\neq \perp$, 0 else }

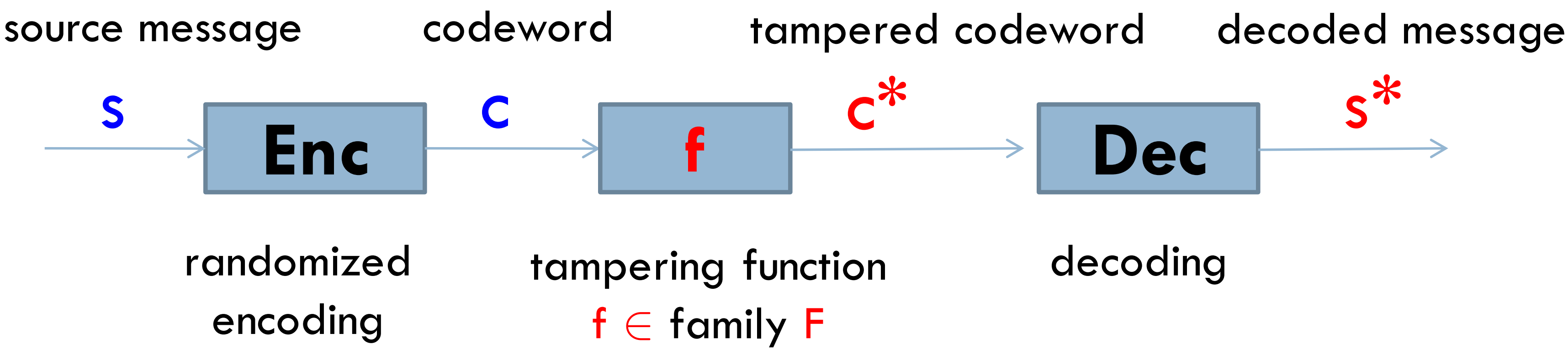
Tamper Detection: Limitations

- Tamper detection fails for functions with many fixed points, or low entropy.

- This is inherent, but perhaps not so bad.
  - Fixed-points: nothing changes!
  - Low-entropy: not much remains!

- Can we relax tamper-detection and still get meaningful security?

Non-Malleability [Dziembowski-Pietrzak-W10]



- <u>Non-Malleability:</u> either $s^* = s$ or s^* is "unrelated" to s.
  - Analogous to non-malleability in cryptography [DDN91].

- Harder to define formally (stay tuned).
- Examples of "malleability":
  - The value s^* is same as s, except with 1st bit flipped.
  - If s begins with 0, then $s^* = s$. Otherwise $s^* = \perp$.

Defining Non-Malleability

<u>Definition:</u> A code (Enc, Dec) is (F, ε)-non-malleable if $\forall f \in F \quad \exists$ simulator Sim_f that outputs an *identity* or a *constant* function g such that $\forall$ s:

$c \leftarrow Enc(s)$, $c^* \leftarrow f(c)$ Output $s^*=Dec(c^*)$	$\approx_\varepsilon$	$g \leftarrow Sim_f$ Output $s^*= g(s)$

General Results for Non-Malleability

- For every code (Enc, Dec) there exists a bad function f, for which the scheme is malleable.
  - f(c) = Enc(Dec(c) + 1).
  - Bad f depends heavily on (Enc, Dec).

Theorem [DPW10, CG13, FMVW14, JW15]:
For any function family F over n-bit codewords, there is an *non-malleable code* for F as long as $|F| < 2^{2^{\alpha n}}$ for $\alpha < 1$.
- Rate of code is $\approx 1 - \alpha$
- If $|F| = 2^{\mathrm{poly}(n)}$ then code can be made efficient.

General Results for Non-Malleability

- Same construction for non-malleable codes and tamper detection. Combine "weak tamper detection" and "leakage resilient" codes: Enc(s) = wtdEnc(LREnc(s)).

- Intuition: few possible outcomes of tampering codeword c.
  - Tamper detection succeeds: $\perp$
  - fixed point f(c) = c: "same"
  - low entropy value f(c) = c' has many pre-images: Dec(c')

- Can think of this as small leakage on LREnc(s).

Much Recent Work

- Explicit efficient constructions:
  - Bit-wise tampering [DPW10,CG13]: each bit of codeword is tampered independently but arbitrarily.
  - Permuting bits of codeword [AGM+14]
  - Split-state model [DKO13, ADL13, ADKO15,CGL15] : Codeword split into two parts that are tampered independently but arbitrarily.
- Applications:
  - CCA security amplification [AGM+14,CMT+15,CDT+15]
  - Non-malleable commitments from OWFs [GPR15]

Application to Tamper-Resilient Security

- Non-malleable codes can protect physical devices against tampering attacks.
  - Store data s on a device in encoded form Enc(s)
  - Each time device is invoked: decode, compute, re-encode

- Tampering of Enc(s) can be simulated by either leaving the data unchanged, or completely overwriting it with a new unrelated value.

- Device has to re-encode the codeword each time with fresh randomness. Is this necessary?

Continuous Tampering and Re-Encoding

- Non-malleable codes only consider one tampering attack per codeword. Can we allow continuous tampering of a single codeword?
- Continuous non-malleable codes (4 flavors):
  [FMV+14, JW15]
  - Device can "self-destruct" if tampering detected?
  - "Persistent" tampering?

Continuous Non-Malleable Codes

Few fixed points, High entropy

No Self-Destruct, Non-Persistent

(strongest)

Self-Destruct,
Non-Persistent

No Self-Destruct,
Persistent

High entropy

Few fixed points

Self-Destruct, Persistent

(weakest)

No restrictions on F

Conclusions

- Defined tamper-detection codes and (continuous) non-malleable codes.

- One general construction. Based on probabilistic method, but can be made efficient for “small” function families.

- Open Questions:
  - Explicit constructions of tamper detection codes and non-malleable codes. More families. Simpler. Better rate.
  - More applications.
    - To non-malleable cryptography
    - To other areas?

Thank you!