Cryptographic Validation Beyond Implementation Correctness

Manuel Barbosa, François Dupressoir, Andreas Hülsing, Vincent Laporte, Pierre-Yves Strub

Cryptographic Validation Beyond Implementation Correctness

Manuel Barbosa, François Dupressoir, Andreas Hülsing, Vincent Laporte, Pierre-Yves Strub

The Problem with Evaluating Cryptography

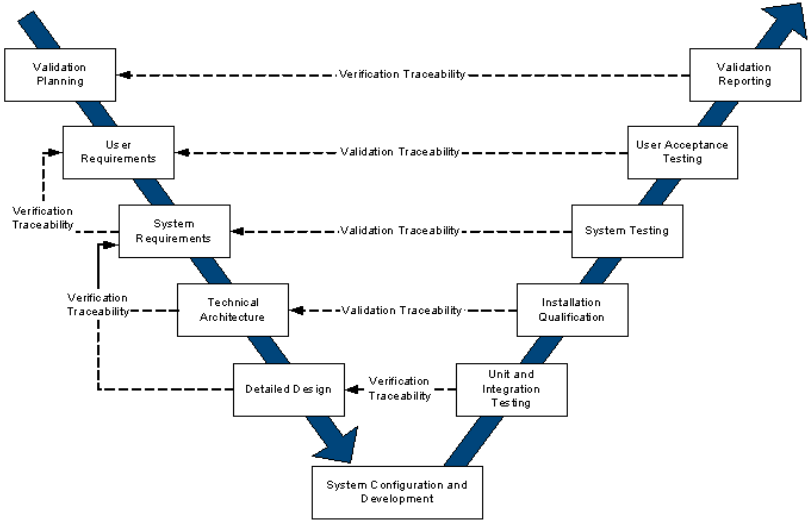
- Cryptographic algorithms
  - Proofs are (typically) published
  - But hard to read, check, *trust*
  - Often apply to a simplified algorithm
- Cryptographic implementations
  - Evaluation is done in private
  - Incentives to obfuscate for evaluation
  - Duplication of expertise

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{Enc}}(A) \leq \mathsf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(B) + \mathsf{Adv}^{\pm\mathsf{prp}}_{\mathsf{E}}(C) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(D) + \frac{14\sigma^2}{2^n} + \frac{2^{n-\tau}}{2^n - 1} + \Pr[\textbf{the proof is wrong}]$$

From Joseph Jaeger, "Adventures in Metacryptography", ProTeCS 2024

Verification for Safety

- (Judicious use of) formal methods moves some defect detection to the left of the V
  - Abstraction and refinement are allies
- *Complement* other verification tools
  - Still need to verify assumptions, …
- *Traceability* is key



Cryptographic Security ≠ Safety

Abstraction, Refinement, and Cryptographic Security

- Abstractions are the adversary's playground

Plaintext Recovery Attacks Against SSH

Martin R. Albrecht, Kenneth G. Paterson and Gaven J. Watson

[Acme] Signature misuse vulnerability in draft-barnes-acme-04

Andrew Ayer <agwa@andrewayer.name> Tue, 11 August 2015 15:54 UTCShow header

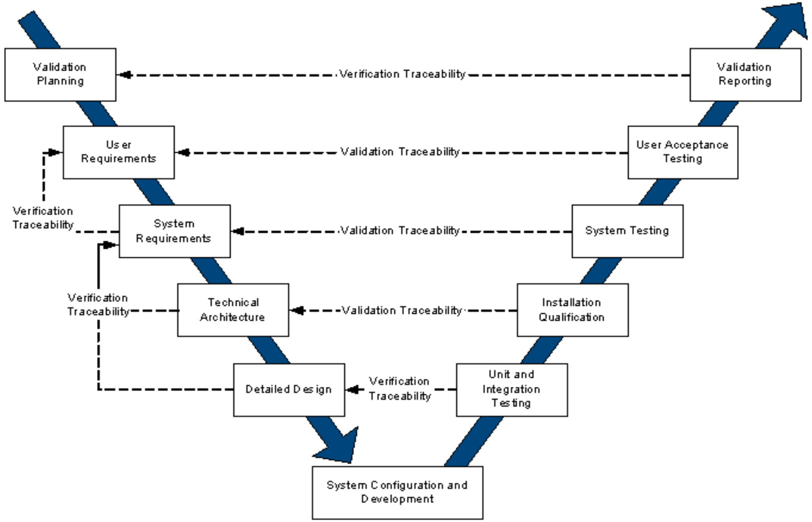
Hash Gone Bad:
Automated discovery of protocol attacks that exploit hash function weaknesses

Vincent Cheval[¶] *Cas Cremers*[‡] *Alexander Dax*[‡]
Lucca Hirschi[†] *Charlie Jacomme*[¶] *Steve Kremer*[*]

- Refinement steps refine *both* the object under study *and* the context in which it is deployed

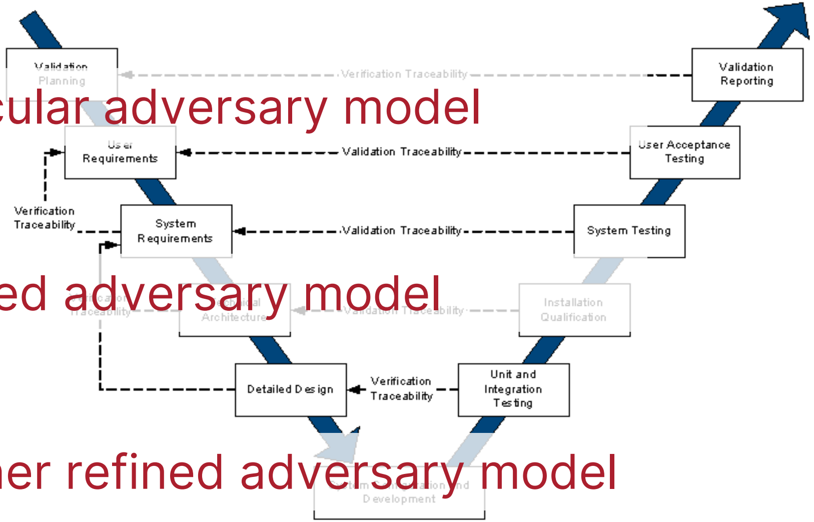
Verification for Cryptographic Security

- *Specify* the algorithm/protocol *and* its expected properties
  - *Verify* security properties
- *Refine* the algorithm into an implementable specification
  - *Verify* security properties by refinement
- *Implement* the specification
  - *Verify* security properties by refinement

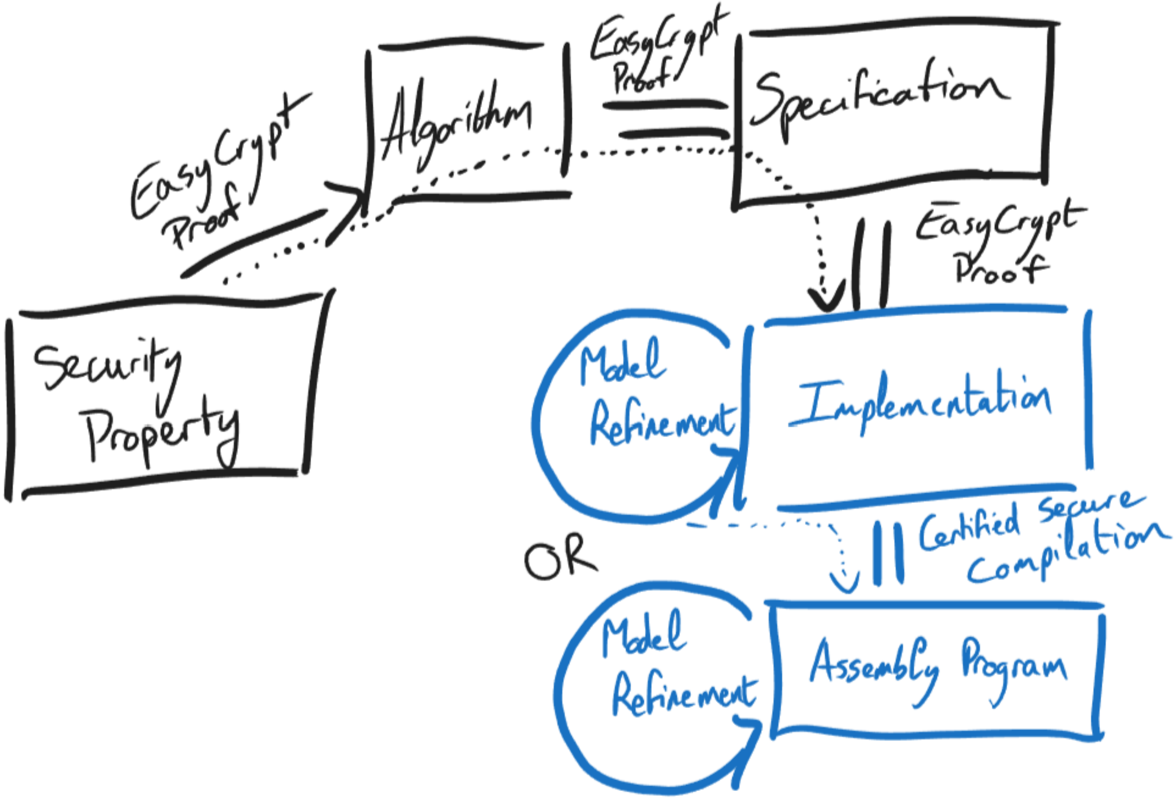


Verification for Cryptographic Security

- *Specify* the algorithm/protocol *and* its expected properties
  - – *Verify* security properties in a particular adversary model
- *Refine* the algorithm into an implementable specification
  - – *Verify* security properties in a refined adversary model by refinement
- *Implement* the specification
  - – *Verify* security properties in a further refined adversary model by refinement



Interlude: The Formosa Crypto Way



Specifying Expected Security is Important for the Verification of Cryptography

- Refinement decisions *must* be informed by expected security
  - Is it fine to add fragmentation?
    It depends on the expected security property.
  - Is it fine to use a Merkle-Damgård hash function here?
    It depends on the expected security property.
  - Is it fine to use a signature scheme that does not bind its public key?
    It depends on the expected security property.
- Also below this!
  - Is it fine to leak this secret-dependent value (through side-channels)?

On the Role of Standards in Verification

- Algorithmic description

Algorithm 2 $\mathsf{Kyber.CPA.Enc}(pk = (\mathbf{t}, \rho), m \in \mathcal{M})$: encryption

$r \leftarrow \{0,1\}^{256}$
$\mathbf{t} := \mathsf{Decompress}_q(\mathbf{t}, d_t)$
$\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \mathsf{Sam}(r)$
$\mathbf{u} := \mathsf{Compress}_q(\mathbf{A}^T \mathbf{r} + \mathbf{e}_1, d_u)$
$v := \mathsf{Compress}_q(\mathbf{t}^T \mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
return $c := (\mathbf{u}, v)$

For verification to make sense, both are needed, and the security of the specification *must* be verified to follow from the security of the algorithm.

- Specification

Algorithm 5 $\textsc{Kyber}.\mathsf{CPAPKE.Enc}(pk, m, r)$: encryption

Input: Public key $pk \in \mathcal{B}^{d_t \cdot k \cdot n/8+32}$
Input: Message $m \in \mathcal{B}^{32}$
Input: Random coins $r \in \mathcal{B}^{32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
$N := 0$
$\mathbf{t} := \mathsf{Decompress}_q(\mathsf{Decode}_{d_t}(pk), d_t)$
$\rho := pk + d_t \cdot k \cdot n/8$
for i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k \times k}$ in NTT domain
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}^T[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho \| i \| j))$
  end for
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{r} \in R_q^k$ from B_η
  $\mathbf{r}[i] := \mathsf{CBD}_\eta(\mathsf{PRF}(r, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e}_1 \in R_q^k$ from B_η
  $\mathbf{e}_1[i] := \mathsf{CBD}_\eta(\mathsf{PRF}(r, N))$
  $N := N + 1$
end for
$e_2 := \mathsf{CBD}_\eta(\mathsf{PRF}(r, N))$ ▷ Sample $e_2 \in R_q$ from B_η
$\hat{\mathbf{r}} := \mathsf{NTT}(\mathbf{r})$
$\mathbf{u} := \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ ▷ $\mathbf{u} := \mathbf{A}^T \mathbf{r} + \mathbf{e}_1$
$v := \mathsf{NTT}^{-1}(\mathsf{NTT}(\mathbf{t})^T \circ \hat{\mathbf{r}}) + e_2 + \mathsf{Decode}_1(\mathsf{Decompress}_q(m, 1))$ ▷ $v := \mathbf{t}^T \mathbf{r} + e_2 + \mathsf{Decompress}_q(m, 1)$
$c_1 := \mathsf{Encode}_{d_u}(\mathsf{Compress}_q(\mathbf{u}, d_u))$
$c_2 := \mathsf{Encode}_{d_v}(\mathsf{Compress}_q(v, d_v))$
return $c = (c_1 \| c_2)$ ▷ $c := (\mathsf{Compress}_q(\mathbf{u}, d_u), \mathsf{Compress}_q(v, d_v))$