Fast verification of masking schemes in characteristic two

Pierre Karpman
Joint work with Nicolas Bordes

Université Grenoble Alpes, France

Séminaire CASC — Grenoble
2020–02–20

Masking schemes for finite field multiplication

Proving security

Computationally checking security in $\mathbb{F}_2$

Applications

Masking schemes for finite field multiplication

Proving security

Computationally checking security in $\mathbb{F}_2$

Applications

The context

Context: Crypto implementation on observable devices

Objective: secure finite-field multiplication w/ leakage

- Implement $(a, b) \mapsto c = a \times b$, a, b, $c \in \mathbb{K}$
  - Used in non-linear ops in sym. crypto (e.g. S-boxes)
  - Input/outputs usually secret!
- Problem: computations leak information
- $\rightsquigarrow$ Need a way to compute a product w/o leaking (too much) the operands & the result
- Our focus: higher-order (many shares) *gadgets*

Basic idea

- Split a, b, c into *shares* (i.e. use a secret-sharing scheme)
  - Typically simple and additive: $x = \sum_{i=0}^{d} x_i$, $x_{0,\ldots,d-1} \xleftarrow{\$} \mathbb{K}$, $x_d = x - \sum_{i=0}^{d-1} x_i$
- Compute the operation over the shared operands; obtain a shared result
- Ensure that neither of a, b, c can be (easily) recovered (e.g. with fewer than $d + 1$ *probes*)

Prove security e.g. in:

- The probing model $\rightsquigarrow$ d-privacy (Ishai, Sahai & Wagner, 2003) / d-(S)NI (Belaïd et al., 2016)
- The noisy leakage model (Chari et al. '99, Prouff & Rivain, 2013)
- (Reductions exist, cf. Duc et al., 2014, 2015)

First attempt

- We want to compute $c = \sum_k c_k = \sum_i a_i \times \sum_j b_j = \sum_{i,j} a_i b_j$
- So maybe define $c_i = \sum_{j=0}^{d} a_i b_j$?
- Problem: any single c_i reveals information about b
- One solution (ISW, 2003): find better partitions and rerandomize using fresh masks
- Prove security in the probing model
- ‽ Scheduling of the operations is important (impacts the probes available to the adversary)

Masking complexity

- ISW provides a practical solution for masking a multiplication
- But the cost is quadratic in d: d-privacy requires:
  - $2d(d+1)$ sums
  - $(d+1)^2$ products
  - $d(d+1)/2$ fresh random masks
- Decreasing the cost/overhead of masking is a major problem
  - Use block ciphers that need few multiplications (e.g. ZORRO, Gérard et al., 2013 (broken))
  - Amortize the cost of masking several mult. (e.g. Coron et al., 2016)
  - Decrease the cost of masking a single mult. (e.g. Belaïd et al., 2016, 2017)

Quick defs.

Gadget

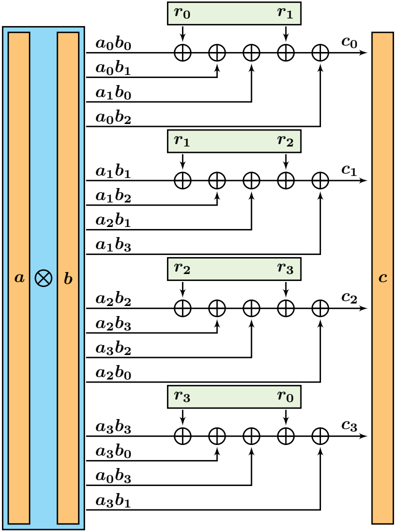
A gadget for a function f is a (randomized) circuit C on (additively) shared intput/outputs $\boldsymbol{x}_i$, $\boldsymbol{y}_j$ s.t. for every set of coins $\mathcal{R}$, $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) \leftarrow C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \mathcal{R})$ satisfies:

$$\left(\sum_{j=1}^{v} \boldsymbol{y}_{1,j}, \ldots, \sum_{j=1}^{v} \boldsymbol{y}_{m,j}\right) = f\left(\sum_{j=1}^{u} \boldsymbol{x}_{1,j}, \ldots, \sum_{j=1}^{u} \boldsymbol{x}_{m,j}\right)$$

Probe

A probe on C maps a wire to the value it takes in a run of the circuit

A 3-NI multiplication gadget (Barthe et al., 2017)



What about today?

- An extension to $\mathbb{F}_2$ of the matrix model (Belaïd et al., 2017) for proving ((S)NI) security
- An efficient algorithm (& implementation) for testing high-order security
- New variants of high-order multiplication gadgets with reduced randomness complexity

Masking schemes for finite field multiplication

Proving security

Computationally checking security in $\mathbb{F}_2$

Applications

A composable security model

- The ISW d-privacy model is not composable: if C_1 and C_2 are d-private, $C_2 \circ C_1$ isn't necessarily so
- Barthe et al. (2016) introduced composable alternatives of *(strong) non-interference*
- Use *simulation-based* definitions
- Roughly, $\mathcal{P} := \{p_1, p_2, \ldots\}$ on C is t-simula(ta)ble if for a fixed input $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots)$, all the distributions induced on $\mathcal{P}$ by $\mathcal{R}$ can be simulated with the knowledge of $\leq t$ $\boldsymbol{x}_{1,i}$s; $\leq t$ $\boldsymbol{x}_{2,i}$s; etc.
- Then, C is d-NI iff. any set of at most d probes is $t \leq d$-simulable
- C is d-SNI iff. any set of at most $d_1 + d_2 \leq d$ probes is d_1-simulable, where d_2 probes are on the output wires only
- And now $SNI \circ NI = SNI$

Examples

- $\boldsymbol{x}_{1,1} + \boldsymbol{r}_1$ is 0-simulable
- $\boldsymbol{x}_{1,1} + \boldsymbol{x}_{2,1}$ is 1-simulable (and not 0-...)
- $\{\boldsymbol{x}_{1,1} + \boldsymbol{x}_{1,2} + \boldsymbol{x}_{1,3} + \boldsymbol{r}_1,\ \boldsymbol{r}_1\}$ is 3-simulable (and not $0, 1, 2$-...)

A matrix model for non-interference

- Now only consider gadgets with (at most) two inputs $\boldsymbol{a}$, $\boldsymbol{b}$ (e.g. for multiplication)
- With only *bilinear probes* (i.e. affine functions of the $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, $\boldsymbol{a}_i\boldsymbol{b}_j$, $\boldsymbol{r}_k$)
- Then it is enough to consider linear combinations of the probes to (dis-)prove (S)NI security (Belaïd et al., 2017; This work)

A linear condition on bilinear probes

Condition 3.2 (Belaïd et al., 2017)

A set of bilinear probes $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ on a circuit C for a function $f : \mathbb{K}^2 \to \mathbb{K}$ satisfies Cond. 3.2 iff. $\exists\, \boldsymbol{\lambda} \in \mathbb{K}^\ell$, $\boldsymbol{M} \in \mathbb{K}^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu} \in \mathbb{K}^{d+1}$, and $\tau \in \mathbb{K}$ s.t. $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i = \boldsymbol{a}^t \boldsymbol{M} \boldsymbol{b} + \boldsymbol{a}^t \boldsymbol{\mu} + \boldsymbol{b}^t \boldsymbol{\nu} + \tau$ and all the rows of the block matrix $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{\mu} \end{pmatrix}$ or all the columns of the block matrix $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{\nu}^t \end{pmatrix}$ are non-zero

- No $\boldsymbol{r}$-dependency: cannot be simulated with a uniform distribution
- No zero rows/columns: full functional dependence on the $d+1$ shares of $\boldsymbol{a}/\boldsymbol{b}$

Proving security with Cond 3.2

The previous condition is useful to analyse the security of a gadget

Theorem (Belaïd et al., 2017)

If $\mathcal{P}$ satisfies Cond. 3.2, then it is not d-simulable
If $\mathcal{P}$ is not d-simulable *and* $\#\mathbb{K} > d + 1$, then it satisfies Cond 3.2

Corollary (Belaïd et al., 2017)

If $\#\mathbb{K} > d + 1$ and no set of $\leq d$ probes on C satisfies Cond. 3.2, then it is d-NI

Not d-sim $\Rightarrow$ Cond 3.2 (sketch)

Let $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ be not d-simulable

- Compute an equiv. set $\mathcal{P}' = \{p'_1 = \sum_i \alpha_i p_i, \ldots\}$ that does not depend on any $\boldsymbol{r}$ and that is not d-simulable (always possible using Gaussian elimination)
- So the matrix $\boldsymbol{D} = \begin{pmatrix} \boldsymbol{M}'_1 & \boldsymbol{\mu}_1 & \cdots \boldsymbol{M}'_{\ell'} & \boldsymbol{\mu}_{\ell'} \end{pmatrix}$ that records the dependence of $\mathcal{P}'$ on say, the $\boldsymbol{a}_i$s has no zero row
- Show that $\exists\, \boldsymbol{\Lambda}$ (encoding a linear comb. of the p's) s.t. $\boldsymbol{D\Lambda}$ has no zero row
- Guaranteed to exist if $\#\mathbb{K} > d + 1$ by Schwartz-Zippel-DeMillo-Lipton (need a non-root of a (degree $d + 1$)-(ℓ'-variate) polynomial)

Why this fails for $\mathbb{F}_2$

Take:

- $d + 1 = 3$
- $p_1 = \boldsymbol{a}_0\boldsymbol{b}_0 + \boldsymbol{a}_1\boldsymbol{b}_0$
- $p_2 = \boldsymbol{a}_1\boldsymbol{b}_0 + \boldsymbol{a}_2\boldsymbol{b}_1$

Then neither p_1, p_2, nor $p_1 + p_2$ depends on all of $\boldsymbol{a}_0\boldsymbol{b}_*$, $\boldsymbol{a}_1\boldsymbol{b}_*$, $\boldsymbol{a}_2\boldsymbol{b}_*$ (so their respective matrix encodings *have* a zero row) but it is obvious that $\{p_1, p_2\}$ is not 2-simulable

- But see that (e.g.) p_1 could be alternatively completed by any $\boldsymbol{a}_2\boldsymbol{b}_*$ (which are always available) to be an attack satisfying Cond 3.2
- In fact any linear comb. of ℓ probes that is not ℓ-simulable is already an attack! (matches the *TNI* variant of NI security)

An alternative condition

Condition 3.2'

A set of bilinear probes $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ on a circuit C for a function $f : \mathbb{K}^2 \to \mathbb{K}$ satisfies Cond 3.2' iff. $\exists\, \boldsymbol{\lambda} \in \mathbb{K}^\ell$, $\mathrm{wt}(\boldsymbol{\lambda}) = \ell$, $\boldsymbol{M} \in \mathbb{K}^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu} \in \mathbb{K}^{d+1}$, and $\tau \in \mathbb{K}$ s.t. $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i = \boldsymbol{a}^t \boldsymbol{M} \boldsymbol{b} + \boldsymbol{a}^t \boldsymbol{\mu} + \boldsymbol{b}^t \boldsymbol{\nu} + \tau$ and the block matrix $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{\mu} \end{pmatrix}$ (resp. the block matrix $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{\nu}^t \end{pmatrix}$) has at least $\ell + 1$ non-zero rows (resp. columns)

Theorem

If a set $\mathcal{P}$ of $\leq d$ bilinear probes on a circuit C for a function $f : \mathbb{K}^2 \to \mathbb{K}$ is not d-simulable then $\exists\, \mathcal{P}' \subseteq \mathcal{P}$ s.t. $\mathcal{P}'$ satisfies Cond 3.2'

Proof (sketch)

- Compute $\{p'_1, \ldots\}$ as before
- Each p'_i has a "p" weight $\leq \ell \leq d$ and an "$\boldsymbol{a}$" weight $\leq d + 1$
- Show that $\exists$ a linear combination of the p's with p weight $< \boldsymbol{a}$ weight $\rightsquigarrow$

Lemma

Let $\mathcal{C}_1$, $\mathcal{C}_2$ be $[n_1, k]$, $[n_2 > n_1, k]$ linear codes over $\mathbb{K}$ generated by $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ w/o zero columns, then the concatenated code $\mathcal{C}_{1,2}$ generated by $\begin{pmatrix} \boldsymbol{G}_1 & \boldsymbol{G}_2 \end{pmatrix}$ is s.t. $\exists\, \boldsymbol{c} \in \mathcal{C}_{1,2}$ w/ $\mathrm{wt}_1(\boldsymbol{c}) < \mathrm{wt}_2(\boldsymbol{c})$, where $\mathrm{wt}_1(\cdot)$ (resp. $\mathrm{wt}_2(\cdot)$) is the weight on the first n_1 (resp. last n_2) coordinates

Proof: by induction, on (appropriately, iteratively) shortened codes

Summary

- Got an easy-to check condition to prove NI security (and not only detect attacks) even over $\mathbb{F}_2$
- (Not shown here) Easy to adapt to prove SNI security (not explicit in previous work)
- (Not shown here) Easy proof that a secure scheme over $\mathbb{F}_2$ can be securely lifted to $\mathbb{F}_{2^n}$ (not explicit in previous work)
- (Not shown here) Can be adapted to *robust probing* (Faust et al., 2018) to take *glitches* into account (currently quite wasteful)

An immediate algorithm from Cond 3.2'

From now on $\mathbb{K} = \mathbb{F}_2$

To prove the d-NI security of a gadget/circuit C (*with only bilinear probes*):

- List all the possible probes $\mathscr{P}$ on C
- For every $\mathcal{P} \in \wp(\mathscr{P})$ of size $\leq d$, check that no full-weight linear combination of all elems of $\mathcal{P}$ satisfies Cond 3.2'
  - Over $\mathbb{F}_2$, this is just $\sum_{p \in \mathcal{P}} p$
- Simple; costs $\sum_{i=1}^{d} \binom{\#\mathscr{P}}{i}$ vector additions

✒Can be seen as a derandomized, proved variant of a heuristic algorithm from Belaïd et al., 2016

Reducing cost with dimension reduction

- Available probes typically include "elementary ones", viz. $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, $\boldsymbol{a}_i\boldsymbol{b}_j$, $\boldsymbol{r}_k$
- It is easy to tell if an existing linear comb. of probes can be completed to an attack using elementary probes
  - E.g. a sum of $< d$ probes that depends on all $\boldsymbol{a}_i$s and exactly one $\boldsymbol{r}_k$
- So remove elementary probes from $\mathscr{P}$ and check a modified Cond 3.2'

✒Already used (except for $\boldsymbol{r}_k$) by Belaïd et al., 2016

Dimension reduction (cont.)

- Concrete gadgets may induce (non-elementary) probes that are always "better" than others
  - E.g. $\boldsymbol{a}_0\boldsymbol{b}_0 + \boldsymbol{r}_0 + \boldsymbol{a}_0\boldsymbol{b}_1 \leq \boldsymbol{a}_0\boldsymbol{b}_0 + \boldsymbol{r}_0 + \boldsymbol{a}_0\boldsymbol{b}_1 + \boldsymbol{a}_1\boldsymbol{b}_0$
  - (But $\boldsymbol{a}_0\boldsymbol{b}_0 + \boldsymbol{r}_0 + \boldsymbol{a}_0\boldsymbol{b}_1 + \boldsymbol{a}_1\boldsymbol{b}_0 \not\gtrless \boldsymbol{a}_0\boldsymbol{b}_0 + \boldsymbol{r}_0 + \boldsymbol{a}_0\boldsymbol{b}_1 + \boldsymbol{a}_1\boldsymbol{b}_0 + \boldsymbol{r}_1$)
- So can reduce dimension further by removing the less useful ones
- Formalizing a sufficient condition + checking that an explicit filtering is valid is not too hard

Efficient software implementation

Implementing the verification is straightforward. For all potential attack set $\in \wp(\mathscr{P})$ of weight $\leq d$:

1. Sum the indicator matrices that encode the probes dependence on $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{r}$
2. Check (the appropriate variant of) Cond 3.2', i.e. compute a block Hamming weight and compare it to a threshold

To make this (a bit? a lot?) more efficient than a naïve implem:

- Use combination Gray codes for the enumeration
- Use vectorization to compute the sums & weights
- $\rightsquigarrow$ peak performance (@2.60 GHz) of $\approx 2^{27.5}$ checks/s

Also, use parallelization

Combination Gray codes

- Enumerate every element of $\{x \in \wp(\mathscr{P}) : \mathrm{wt}(x) = k\}$ as a sequence $x_1, \ldots$ s.t. $\# x_i \backslash x_{i+1} = 2$
- So can compute $\sum_{v \in x_{i+1}} v$ from $\sum_{v \in x_i} v$ using one addition and one subtraction (independent of k)
- Several codes with this property exist; we use the "Nijenhuis-Wilf-Tang-Liu" one whose combinations have easy-to-compute (un)ranking maps to and from $\mathbb{N}$
  - So easy to split a search space for a parallel implementation

Vectorized block Hamming weight with AVX512VL + AVX512BW

Pretty easy up to $d + 1 = 16$:

```
int popcount256_16(__m256i v)
{
        return
         ↪  __builtin_popcountl(_mm256_cmpgt_epi16_mask(
         ↪  v, _mm256_setzero_si256()));
}
```

Use several words for larger cases (too expensive to run till the end anyways)

Masking schemes for finite field multiplication

Proving security

Computationally checking security in $\mathbb{F}_2$

Applications

Why are you doing this?

- Initial goal: prove the security at high-order of “new” multiplication gadgets over $\mathbb{F}_2$ w/ reduced randomness complexity
- Turns out those were already proposed by Barthe et al. in 2017 :((but we have better variants sometimes)

Soooo... what’s left?

- Beats state-of-the-art verification performance of multiplication gadgets by three orders of magnitude
- Disprove a generalization conjecture from Barthe et al. (2017)
- Verify (S)NI multiplication up to order 11 (up from 7)
- Still some improvements, e.g. 17% (resp. 19%) randomness gain for 8-share SNI multiplication (resp. refreshing)

Verification performance

For one 8-SNI multiplication gadget:

- The latest version of maskVerif (Barthe et al., 2019) takes 13 days on up to 4 threads to prove security
- Our software does it in < 10 minutes on 1 thread

For one 11-NI multiplication gadget:

- maskVerif: not run...
- Our software: wall time of three days; used up to 16 nodes of the *Dahu* cluster ($\Rightarrow$ up to 512 cores) to enumerate $\approx 2^{52.72}$ possible attack sets (down from 2^{59} before non-elementary filtering)

SNI security is hard

Roughly, to get SNI security:

- Start from an NI-secure scheme
- Add refreshing before the output

SNI security is hard

6-NI:

```
s00 r00 s01 s10 r01 s02 s20 r07 s03 s30 r08
s11 r01 s12 s21 r02 s13 s31 r08 s14 s41 r09
s22 r02 s23 s32 r03 s24 s42 r09 s25 s52 r10
s33 r03 s34 s43 r04 s35 s53 r10 s36 s63 r11
s44 r04 s45 s54 r05 s46 s64 r11 s40 s04 r12
s55 r05 s56 s65 r06 s50 s05 r12 s51 s15 r13
s66 r06 s60 s06 r00 s61 s16 r13 s62 s26 r07
```

SNI security is hard

6-SNI:

```
s00 r00 s01 s10 r01 s02 s20 r07 s03 s30 r08 r14 r20
s11 r01 s12 s21 r02 s13 s31 r08 s14 s41 r09 r15 r14
s22 r02 s23 s32 r03 s24 s42 r09 s25 s52 r10 r16 r15
s33 r03 s34 s43 r04 s35 s53 r10 s36 s63 r11 r17 r16
s44 r04 s45 s54 r05 s46 s64 r11 s40 s04 r12 r18 r17
s55 r05 s56 s65 r06 s50 s05 r12 s51 s15 r13 r19 r18
s66 r06 s60 s06 r00 s61 s16 r13 s62 s26 r07 r20 r19
```

SNI security is hard

Roughly, to get SNI security:

- Start from an NI-secure scheme
- Add refreshing before the output
- So Barthe et al. (2017) conjectured that a single refreshing as above was always enough
- We did too ("independently", 3 years after...); checked; it fails from $d = 10$ if a rotation by *one* is used for the refreshing
  - Yet, used as is in the $d + 1 \in \{16, 32\}$ implementations by Journault and Standaert (2017)??
- But a rotation by *two* works there... always the case?
- It's also often possible to add even less, e.g. 4 masks (instead of 8) for $d = 7 \leftarrow$ one of our improvements!

State-of-the-art 7-SNI multiplication

7-SNI multiplication with 20 masks:

```
s00 r00 s01 s10 r01 s02 s20 r08 s03 s30 r09 s04 r20
s11 r01 s12 s21 r02 s13 s31 r09 s14 s41 r10 s15 r21
s22 r02 s23 s32 r03 s24 s42 r10 s25 s52 r11 s26 r22
s33 r03 s34 s43 r04 s35 s53 r11 s36 s63 r12 s37 r23
s44 r04 s45 s54 r05 s46 s64 r12 s47 s74 r13 s40 r20
s55 r05 s56 s65 r06 s57 s75 r13 s50 s05 r14 s51 r21
s66 r06 s67 s76 r07 s60 s06 r14 s61 s16 r15 s62 r22
s77 r07 s70 s07 r00 s71 s17 r15 s72 s27 r08 s73 r23
```

References

- Preprint: https://eprint.iacr.org/2019/1165 (currently out of date)
- Implementation: https://github.com/NicsTr/binary_masking