Efficient Actively Secure OT Extension: 5 Years Later[1] (Part I)

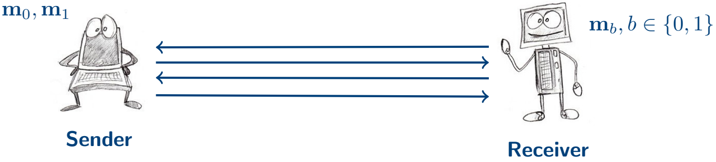
Emmanuela Orsini and Peter Scholl

imec-COSIC, KU Leuven and Aarhus University

[1] Based on the paper *Efficient Actively Secure OT Extension*, M. Keller, E. Orsini, P. Scholl CRYPTO 2015

Oblivious transfer - Definition

Oblivious Transfer (OT) is a ubiquitous cryptographic primitive designed to transfer specific data based on the receiver's choice.

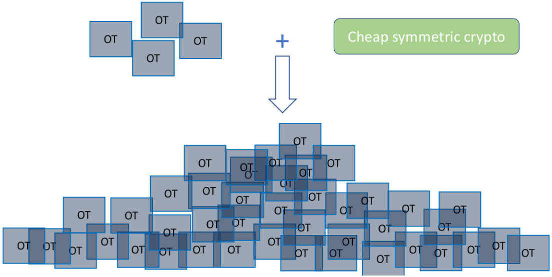


No further information should be learned by any party

Relevant to this workshop: distribution of keys for GC, Threshold ECDSA, etc..

Extending oblivious transfer - Motivation

- Impagliazzo, Rudich [IR98]
  Black-box separation result $\rightarrow$ OT is impossible without public-key primitives (?)
- Beaver [Beaver96]: OT can be extended



OT-extension: 2003-2020

- **Y. Ishai, J. Kilian, K. Nissim, E. Petrank**
  "Extending oblivious transfers efficiently", CRYPTO 2003
- G. Asharov, Y. Lindell, T. Schneider, and M. Zohner
  More Efficient Oblivious Transfer and Extensions for Faster Secure Computation, ACM CCS 2013
- V. Kolesnikov, R. Kumaresan
  Improved OT extension for transferring short secrets, CRYPTO 2013

+ J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra.
  A new approach to practical active-secure two-party computation, CRYPTO 2012
+ G. Asharov, Y. Lindell, T. Schneider, and M. Zohner
  More efficient oblivious transfer extensions with security for malicious adversaries, EUROCRYPT 2015
+ **M. Keller, E. Orsini, P. Scholl**
  Actively Secure OT Extension with Optimal Overhead, CRYPTO 2015
+ **M. Orrù, E. Orsini, P. Scholl**
  Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection, CT-RSA 2017

x D. Masny, P. Rindal
  Endemic Oblivious Transfer, CCS 2019

x C. Guo, J. Katz, X. Wang, Y. Yu
  Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers, IEEE S&P 2020

* **E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Scholl Efficient Pseudorandom Correlation Generators: Silent OT Extension and More, CRYPTO 2019**

OT, Correlated OT and Random OT

Standard OT and COT functionality (Sender chosen message)

OT and COT with uniform message security

OT, Correlated OT and Random OT

Standard OT and COT functionality (Sender chosen message)

Endemic security [MR19]

OT, Correlated OT and Random OT

Standard OT and COT functionality (Sender chosen message)

Endemic security [MR19]

IKNP OT-extension

	Receiver 	Sender 
INPUT.	$(x_1, \ldots, x_m) \in \{0,1\}^m$	$\mathbf{m}_{0,i}, \mathbf{m}_{1,i} \in \{0,1\}^k$ $i \in [m], k \ll m$
1. m COT	$\mathbf{t}_i, \mathbf{x}$ $\mathbf{t}_i \in \{0,1\}^k, i \in [m]$	$\mathbf{q}_i, \Delta$ $\mathbf{t}_i = \mathbf{q}_i + x_i \cdot \Delta$
2. RO	$\mathbf{m}_{x_i,i} = H(\mathbf{t}_i, i) + \mathbf{c}_{x_i,i}$	Send: $\mathbf{c}_{0,i} = H(\mathbf{q}_i, i) + \mathbf{m}_{0,i}$ $\mathbf{c}_{1,i} = H(\mathbf{q}_i + \Delta, i) + \mathbf{m}_{1,i}$

IKNP OT extension - Security

- Assuming that Phase 1. of the protocol is passively/actively secure then
  - IKNP is passively/actively secure when H is a random oracle
  - For passive security it is enough for H to be a correlation robust hash function [IKNP03]
  - For active security H has to be a tweakable correlation robust hash function
- To achieve active security we need:
  - Prove that Phase 1 is secure
    1. Achieve security against a malicious receiver
  - Secure instantiation of the building blocks

IKNP OT extension - Security

- Assuming that Phase 1. of the protocol is passively/actively secure then
  - IKNP is passively/actively secure when H is a random oracle
  - For passive security it is enough for H to be a correlation robust hash function [IKNP03]
  - For active security H has to be a tweakable correlation robust hash function
- To achieve active security we need:
  - Prove that Phase 1 is secure
    1. Achieve security against a malicious receiver
  - Secure instantiation of the building blocks

Proctecting against a malicious receiver - Attack

$$\begin{array}{c} \mathbf{q}_1 = \mathbf{t}_1 + x_1 \cdot \Delta \\ \mathbf{q}_2 = \mathbf{t}_2 + x_2 \cdot \Delta \\ \mathbf{q}_3 = \mathbf{t}_3 + x_3 \cdot \Delta \\ \\ \vdots \\ \\ \mathbf{q}_m = \mathbf{t}_m + x_m \cdot \Delta \end{array} \begin{pmatrix} t_{1,1} + x_1 \cdot \Delta_1 & \dots & t^{\kappa}_{1,k} + \cdot x_1 \cdot \Delta_k \\ t_{2,1} + x_2 \cdot \Delta_1 & \dots & t^{\kappa}_{2,k} + x_2 \cdot \Delta_k \\ t_{3,1} + x_3 \cdot \Delta_1 & \dots & t^{\kappa}_{3,k} + tx_3 \cdot \Delta_k \\ \vdots & \dots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \dots & \vdots \\ t_{m,1} + x_m \cdot \Delta_1 & \dots & t_{m,k} + x_m \cdot \Delta_k \end{pmatrix}$$

Protecting against a malicious receiver - Attack

$$\mathbf{q}_1 = \mathbf{t}_1 + (\Delta_1, 0, \ldots, 0)$$

$$\mathbf{q}_2 = \mathbf{t}_2 + (0, \Delta_2, 0, \ldots, 0)$$

$$\mathbf{q}_3 = \mathbf{t}_3 + (0, 0, \Delta_3, 0, \ldots, 0)$$

$$\vdots$$

$$\begin{pmatrix} t_{1,1} + \Delta_1 & \ldots & \ldots & t^{\kappa}_{1,k} \\ t_{2,1} & t_{2,2} + \Delta_2 & \ldots & t^{\kappa}_{2,k} \\ t_{3,1} & \ldots & \ldots & t^{\kappa}_{3,k} \\ \vdots & \vdots & \ldots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ldots & \vdots \\ t_{m,1} & \ldots & \ldots & t_{m,k} + \Delta_k \end{pmatrix}$$

- $\mathbf{c}_{0,1} = H(\mathbf{q}_1, 1) + \mathbf{m}_{0,1} = H(\mathbf{t}_1 + (\Delta_1, 0, \ldots, 0), 1) + \mathbf{m}_{0,1}$, can extract Δ_1
- Repeating the attack can recover the entire Δ and hence all the messages

Protecting against a malicious receiver - Consistency check

	Receiver 	**Sender** 
INPUT	$(x_1, \dots, x_m) \in \{0,1\}^m$ $(x_{m+1}, \dots, x_{m'}) \in \{0,1\}^{m'-m}$, $m' - m = k + s$	$\mathbf{m}_{0,i}, \mathbf{m}_{1,i} \in \{0,1\}^k$ $i \in [m'], k \ll m'$
1. m COT$^-$	$\mathbf{t}_i, \mathbf{x}$ $\mathbf{t}_i \in \{0,1\}^k, i \in [m']$	$\mathbf{q}_i, \Delta$ $\mathbf{q}_i + \mathbf{t}_i = x_i \cdot \Delta$
2. CHECK	Receive $\chi_1, \dots, \chi_{m'} \in \mathbb{F}_{2^k}$ Send $t = \sum_i \chi_i t_i$ and $x = \sum_i \chi_i x_i$	Compute $q = \sum_i \chi_i q_i$ and check that $t = q + x \cdot \Delta$
3. RO	$\mathbf{m}_{x_i,i} = H(\mathbf{t}_i, i) + \mathbf{c}_{x_i,i}$	$\mathbf{c}_{0,i} = H(\mathbf{q}_i, i) + \mathbf{m}_{0,i}$ $\mathbf{c}_{1,i} = H(\mathbf{q}_i + \Delta, i) + \mathbf{m}_{1,i}$

Part II: Instantiating the Primitives; and Silent OT Extension

Instantiating the Base OTs [Masny-Rindal 19]

Some instantiations allow corrupt parties to bias random-OT outputs

- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \xrightarrow{\text{OT-ext}} (\mathsf{COT}^-, \mathsf{ROT}^- \text{ or } \mathsf{OT})$
- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \not\xrightarrow{\text{OT-ext}} \mathsf{ROT}$

Instantiating the Base OTs [Masny-Rindal 19]

Some instantiations allow corrupt parties to bias random-OT outputs

- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \xrightarrow{\text{OT-ext}} (\mathsf{COT}^-, \mathsf{ROT}^- \text{ or } \mathsf{OT})$
- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \not\xrightarrow{\text{OT-ext}} \mathsf{ROT}$

Instantiating the Base OTs [Masny-Rindal 19]

Some instantiations allow corrupt parties to bias random-OT outputs

- (OT or OT$^-$) $\xrightarrow{\text{OT-ext}}$ (COT$^-$, ROT$^-$ or OT)
- (OT or OT$^-$) $\overset{\text{OT-ext}}{\not\longrightarrow}$ ROT

		Receiver 	Sender 
	Input	$x_1 \in \{0,1\}$	
1.	m COT	$\mathbf{t}, x_1 \in \{0,1\}^k$ $\mathbf{t} \in \{0,1\}^k$	$\mathbf{q}, \Delta$ $\mathbf{q} + \mathbf{t} = x_1 \cdot \Delta$
2.	Check		
3.	RO	$\mathbf{m}_{x_1} \; = H(\mathbf{t}, 1)$	$\mathbf{m}_0 = H(\mathbf{q}, 1)$ $\mathbf{m}_1 = H(\mathbf{q} + \Delta, 1)$

Instantiating the Base OTs [Masny-Rindal 19]

Some instantiations allow corrupt parties to bias random-OT outputs

- $(\text{OT or OT}^-) \xrightarrow{\text{OT-ext}} (\text{COT}^-, \text{ROT}^- \text{ or OT})$
- $(\text{OT or OT}^-) \overset{\text{OT-ext}}{\not\longrightarrow} \text{ROT}$

		Receiver 	Sender 
	INPUT	$x_1 \in \{0,1\}$	
1.	m COT	$\mathbf{0} \in \{0,1\}^k$ $\mathbf{0}, x_1 = 1$	$\mathbf{q}, \Delta$ $\mathbf{q} = \Delta$
2.	CHECK		
3.	RO	$\mathbf{m}_1 = H(\mathbf{0}, 1)$	$\mathbf{m}_0 = H(\mathbf{q}, 1)$ $\mathbf{m}_1 = H(\mathbf{0}, 1)$

Instantiating the Base OTs [Masny-Rindal 19]

Some instantiations allow corrupt parties to bias random-OT outputs

- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \xrightarrow{\text{OT-ext}} (\mathsf{COT}^-, \mathsf{ROT}^- \text{ or } \mathsf{OT})$
- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \not\xrightarrow{\text{OT-ext}} \mathsf{ROT}$

- COT^- or ROT^- enough for OT and most applications
  - But not always: e.g. be careful with ROT^- and some PSI protocols

- If true ROT needed, protocols can be modified:

$$\mathsf{OT}^- \xrightarrow{\text{OT-ext}} \mathsf{COT}^- \xrightarrow{\text{coin}} \mathsf{ROT}$$

Instantiating the Base OTs [Masny-Rindal 19]

Some instantiations allow corrupt parties to bias random-OT outputs

- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \xrightarrow{\text{OT-ext}} (\mathsf{COT}^-, \mathsf{ROT}^- \text{ or } \mathsf{OT})$
- $(\mathsf{OT} \text{ or } \mathsf{OT}^-) \not\xrightarrow{\text{OT-ext}} \mathsf{ROT}$

- COT^- or ROT^- enough for OT and most applications
  - But not always: e.g. be careful with ROT^- and some PSI protocols

- If true ROT needed, protocols can be modified:

$$\mathsf{OT}^- \xrightarrow{\text{OT-ext}} \mathsf{COT}^- \xrightarrow{\text{coin}} \mathsf{ROT}$$

Instantiating the hash function $H(x, i)$ [GKWY 20]

Security requirement: form of *correlation robustness*

Instantiating the hash function $H(x, i)$ [GKWY 20]

Security requirement: form of *correlation robustness*

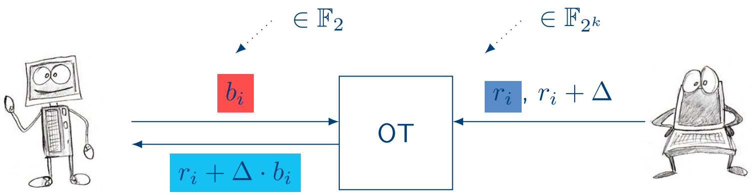
- SHA 256: straightforward, but slow
- Fixed-key block cipher, e.g. AES
  - ≈ 10x faster
  - Incorporating index i: can be done with one extra AES call [GKWY20]

Instantiating the hash function $H(x, i)$ [GKWY 20]

Security requirement: form of *correlation robustness*

- SHA 256: straightforward, but slow
- Fixed-key block cipher, e.g. AES
  - ≈ 10x faster
  - Incorporating index i: can be done with one extra AES call [GKWY20]
- What if i is omitted?
  - Can lead to attack, depending on base OTs [MR19]

Silent OT Extension: a Different Approach to Correlated OT [BCGIKS19]

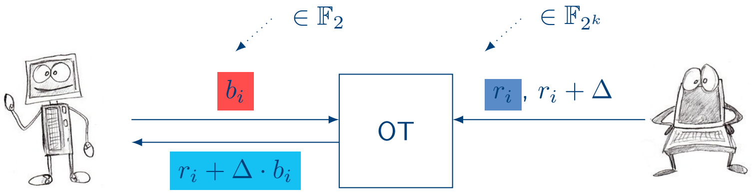


As vectors: variant of vector-OLE with $b_i \in \mathbb{F}_2$

$$\Delta \cdot \mathbf{b} = \mathbf{r} + \Delta \cdot \mathbf{b} + \mathbf{r}$$

Silent OT: compress vector-OLE with a pseudorandom correlation generator (PCG)

Silent OT Extension: a Different Approach to Correlated OT [BCGIKS19]

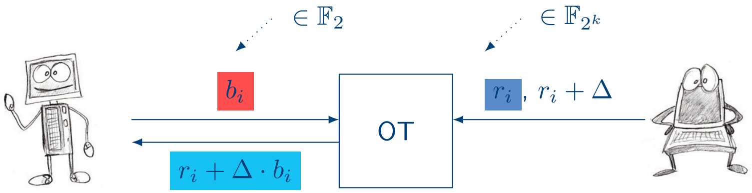


As vectors: variant of vector-OLE with $b_i \in \mathbb{F}_2$

$$\Delta \cdot \boxed{\mathbf{b}} = \boxed{\mathbf{r} + \Delta \cdot \mathbf{b}} + \boxed{\mathbf{r}}$$

Silent OT: compress vector-OLE with a pseudorandom correlation generator (PCG)

Silent OT Extension: a Different Approach to Correlated OT [BCGIKS19]



As vectors: variant of vector-OLE with $b_i \in \mathbb{F}_2$

$$\Delta \cdot \mathbf{b} = \mathbf{r} + \Delta \cdot \mathbf{b} + \mathbf{r}$$

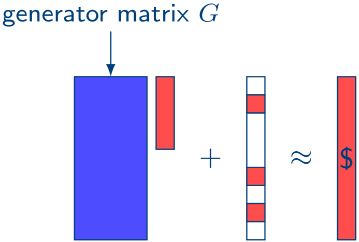
Silent OT: compress vector-OLE with a pseudorandom correlation generator (PCG)

From a PCG to Silent OT Extension

1. Setup protocol for generating keys [BCGIKRS19, SGRR19]
   - 2-round setup for puncturable PRF
2. Malicious security [BCGIKRS19,YWLZW20]
   - Consistency check (similar to [KOS15]), $< 10\%$ overhead

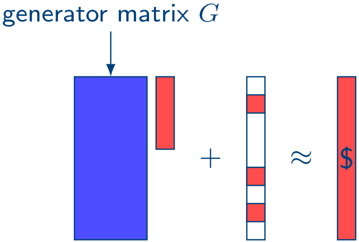
Security of Silent OT: variants of Learning Parity with Noise

Primal-LPN:



Security of Silent OT: variants of Learning Parity with Noise

Primal-LPN:

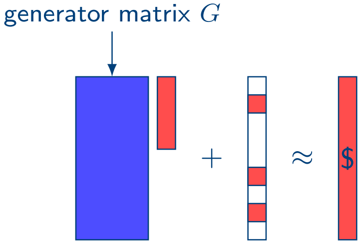


limited to quadratic stretch

G can be *sparse* $\Rightarrow$ faster

Security of Silent OT: variants of Learning Parity with Noise

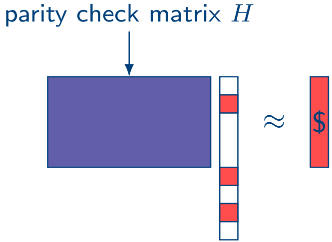
Primal-LPN:



limited to quadratic stretch

G can be *sparse* $\Rightarrow$ faster

Dual-LPN:

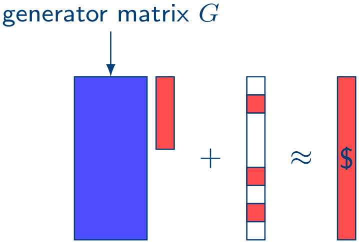


arbitrary polynomial stretch

H must be dense; use *quasi-cyclic codes*

Security of Silent OT: variants of Learning Parity with Noise

Primal-LPN:

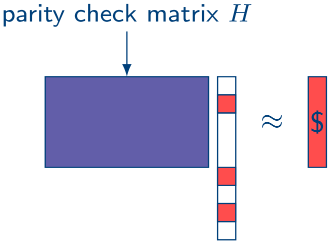


limited to quadratic stretch

G can be *sparse* $\Rightarrow$ faster

Security as in [Ale03]

Dual-LPN:



arbitrary polynomial stretch

H must be dense; use *quasi-cyclic codes*

Security as in BIKE, HQC schemes

Comparing practical, actively secure OT extension protocols

128-bit security; estimates for 10 million random OTs

Reference	Silent	Rounds	Communication	Computation	Based on
[KOS15]	✗	$3/5^*$	160 MB	$\approx 0.2s$	crh
[BCGIKRS19]	✓	$2/4^*$	80 kB	$\approx 2.0s$	QC-reg-LPN, crh
[YWLZW20]	✓	$O(1)$	2.4 MB	≈ 0.3s	sparse-reg-LPN, crh
[YWLZW20]	✓	$O(1)$	2.1 MB	≈ 0.2s	sparse-LPN, crh

* passive/active; crh = correlation robust hash function

Conclusion

- Pitfalls when implementing OT extension
  - Take care with hashing, and security of random OT

- Many flavours of OT extension to choose from:
  - Correlated OT, random OT
  - 1-out-of-2, 1-out-of-N
  - IKNP-style, silent