A Detailed Exposition of the Bellare-Micali 1-out-of-2 Oblivious Transfer Protocol

Debrup Chatterjee

January 30, 2025

Abstract

Oblivious Transfer (OT) is a fundamental cryptographic building block with numerous applications in secure computation and privacy-preserving protocols. The Bellare-Micali 1-out-of-2 OT protocol is a classic and elegant solution under the Decisional Diffie-Hellman (DDH) assumption, leveraging ElGamal-like exponentiation and hashing. In this document, we present a detailed derivation of the protocol, step-by-step mathematical exposition, and an analysis of its security.

Contents

1 Introduction

1.1 Background and Motivation

Oblivious Transfer (OT) stands as one of the most fundamental primitives in modern cryptography, serving as a building block for numerous secure computation protocols. First introduced by Rabin in 1981, and later refined by Even, Goldreich, and Lempel, OT has evolved from a theoretical concept to a practical tool essential for privacy-preserving protocols.

In its most basic form, a 1-out-of-2 OT protocol involves two parties: a *sender* who holds two messages (x_0, x_1), and a *receiver* who wishes to learn exactly one of these messages x_b based on their choice bit $b \in \{0, 1\}$. The protocol must satisfy two crucial security properties:

- **Receiver Privacy:** The sender should remain oblivious to which message was chosen (i.e., the value of b).
- **Sender Privacy:** The receiver should learn only one message x_b and gain no information about the other message x_{1-b}.

1.2 Applications and Significance

The importance of OT extends far beyond its simple description, playing a crucial role in:

- **Secure Multi-party Computation (MPC):** OT serves as the foundation for general secure computation protocols, enabling multiple parties to jointly compute functions while keeping their inputs private.
- **Private Set Intersection (PSI):** Protocols allowing two parties to find common elements in their sets without revealing other elements heavily rely on OT as a building block.

- **Zero-Knowledge Proofs:** Many efficient zero-knowledge protocols utilize OT as a fundamental component to achieve their security properties.
- **Password-Based Authentication:** OT can be used to construct secure password authentication protocols that protect against offline dictionary attacks.

1.3 Historical Development

The evolution of OT protocols represents a fascinating journey in cryptographic research:

- **1981:** Rabin introduced the original concept of OT where the receiver receives a message with probability 1/2.
- **1985:** Even, Goldreich, and Lempel proposed the 1-out-of-2 variant, which proved more practical for applications.
- **1992:** Bellare and Micali presented their DDH-based protocol, offering a simple and elegant solution.
- **Recent Developments:** Modern implementations focus on efficiency improvements through techniques like OT extension and batch processing.

The Bellare-Micali protocol [1] represents a particularly elegant implementation of 1-out-of-2 OT. Its security relies on the well-studied Decisional Diffie-Hellman (DDH) assumption in cyclic groups, making it both theoretically sound and practically implementable. The protocol cleverly combines ElGamal-style encryption with hashing techniques to achieve its security goals.

1.4 Notation and Assumptions

To proceed with the formal description of the protocol, we establish the following notation and assumptions:

- Let p be a large prime, typically chosen to provide security equivalent to standard cryptographic parameters (e.g., 2048 bits for classical security).
- Let G be a cyclic group of order p with a generator g. Common instantiations include:
  - A prime-order subgroup of $\mathbb{Z}_p^*$
  - An elliptic curve group over a finite field
  - Other cryptographic groups where DDH is believed to be hard
- We denote by $H(\cdot)$ a cryptographic hash function that maps elements of G (or bitstrings derived thereof) to $\{0,1\}^n$ for some security parameter n. This function is modeled as a random oracle in security proofs.
- We assume the DDH assumption holds in G. Formally, for random $a, b \in \mathbb{Z}_p$, given (g, g^a, g^b), it is computationally infeasible to distinguish g^{ab} from a random element in G.
- The sender, denoted as A, has two messages: $x_0, x_1 \in \{0,1\}^n$.
- The receiver, denoted as B, has a choice bit $b \in \{0,1\}$ indicating which message they want to learn.

1.5 Security Goals

The protocol aims to achieve several security properties:

- **Correctness:** If both parties follow the protocol, the receiver should successfully obtain their chosen message with probability 1.
- **Receiver Privacy:** The sender should learn nothing about the receiver's choice bit b, even if they deviate from the protocol (malicious behavior).
- **Sender Privacy:** The receiver should learn exactly one message and nothing about the other message, even under malicious behavior.
- **Simulatability:** The protocol should be simulatable in both the semi-honest and malicious security models.

2 Protocol Definition

2.1 High-Level Idea

At a high level, the Bellare-Micali protocol implements oblivious transfer through a clever application of public key cryptography. The core insight lies in creating a pair of public keys with special properties:

- The receiver generates two public keys (PK_0, PK_1) but can only know the secret key for one of them
- The public keys are mathematically related such that their product equals a public value
- To the sender, both public keys appear equally valid and indistinguishable

The protocol achieves this through several elegant steps:

1. The sender first publishes a random group element $c = g^u$
2. For choice bit b, the receiver creates:
   - A "real" public key $PK_b = g^k$ for which they know the secret key k
   - A "fake" public key $PK_{1-b} = c/g^k$ for which they cannot know the secret key
3. This construction ensures $PK_0 \cdot PK_1 = c$, making both keys look valid to the sender
4. The DDH assumption ensures the sender cannot tell which key is "real" and which is "fake"

2.2 Protocol Steps

1. **Setup:**
   - The sender A publishes a random element $c \in G$.
   $$c = g^u \quad \text{for some random } u \in \mathbb{Z}_p.$$
2. **Receiver's Key Generation:**
   - B chooses a random exponent $k \in \mathbb{Z}_p$.
   - B computes
   $$PK_b = g^k, \quad PK_{1-b} = \frac{c}{g^k} = g^u \cdot g^{-k} = g^{u-k}.$$
   - Note that $PK_0 \cdot PK_1 = g^k \cdot g^{u-k} = g^u = c$, verifying the multiplication check.
   - B sends (PK_0, PK_1) to A.
   - A checks that $PK_0 \cdot PK_1 = c$. If the equality fails, A aborts.
3. **Sender's Encryption:**
   - A samples random exponents $r_0, r_1 \in \mathbb{Z}_p$.
   - A then computes two ciphertexts (C_0, C_1):
   $$C_0 = \big(g^{r_0},\, H\big((PK_0)^{r_0}\big) \oplus x_0\big), \quad C_1 = \big(g^{r_1},\, H\big((PK_1)^{r_1}\big) \oplus x_1\big).$$
   - A sends (C_0, C_1) to B.
4. **Receiver's Decryption:**
   - Upon receiving (C_0, C_1), B decrypts the ciphertext corresponding to its choice bit b:
   $$C_b = (V_1, V_2) = \big(g^{r_b},\, H((PK_b)^{r_b}) \oplus x_b\big).$$
   - Since $PK_b = g^k$, we have $(PK_b)^{r_b} = (g^k)^{r_b} = g^{kr_b}$.
   - Let $V_1 = g^{r_b}$. Then,
   $$V_1^k = (g^{r_b})^k = g^{kr_b}.$$
   - B recovers
   $$x_b = H\big(g^{kr_b}\big) \oplus \Big(H\big(g^{kr_b}\big) \oplus x_b\Big).$$
   - This simplifies to x_b, as the XOR of y with itself is 0: $y \oplus y = 0$.

2.3 Pseudocode

Algorithm 1 Bellare-Micali 1-out-of-2 Oblivious Transfer

1: **Input:**
- Sender A has messages $x_0, x_1 \in \{0,1\}^n$.
- Receiver B has choice bit $b \in \{0,1\}$.

2: **Output:**
- B learns x_b.
- A remains oblivious to b.

3: **Setup:** A samples a random $u \in \mathbb{Z}_p$ and sets $c = g^u$. A sends c to B.

4: **Receiver's KeyGen:**
- B picks a random $k \in \mathbb{Z}_p$.
- B computes $PK_b = g^k$ and $PK_{1-b} = c \,/\, g^k$.
- B sends (PK_0, PK_1) to A.

5: **Check:** A verifies that $PK_0 \cdot PK_1 = c$. If it holds, proceed; otherwise, abort.

6: **Sender's Encryption:**
- A picks random exponents $r_0, r_1 \in \mathbb{Z}_p$.
- A computes:

$$C_0 \leftarrow \Big(g^{r_0},\ H\big((PK_0)^{r_0}\big) \oplus x_0\Big),$$

$$C_1 \leftarrow \Big(g^{r_1},\ H\big((PK_1)^{r_1}\big) \oplus x_1\Big).$$

- A sends (C_0, C_1) to B.

7: **Receiver's Decryption:**
- B takes $C_b = (V_1, V_2)$.
- B computes $V_1^k = (g^{r_b})^k = g^{kr_b}$.
- B recovers $x_b \leftarrow H\big(g^{kr_b}\big) \oplus V_2$.

8: **Output:** B obtains x_b.

3 Correctness Proof

Theorem 3.1: Correctness

The Bellare-Micali protocol satisfies the following correctness properties:

1. The receiver B successfully learns x_b for its chosen bit b

2. The protocol reveals no additional information about x_{1-b}
3. The decryption process is guaranteed to work when both parties follow the protocol

Proof of Correctness. We prove correctness by analyzing each component of the protocol systematically:

Part 1: Validity of Public Key Construction

First, we verify that the public keys are properly constructed:

- For the chosen bit b, B sets $PK_b = g^k$ where k is known
- The other key is set as $PK_{1-b} = c/g^k = g^u/g^k = g^{u-k}$
- This ensures $PK_0 \cdot PK_1 = g^k \cdot g^{u-k} = g^u = c$

Part 2: Analysis of Encryption

The sender creates two ciphertexts:

$$C_0 = \left(g^{r_0},\, H((PK_0)^{r_0}) \oplus x_0\right) \quad \text{and} \quad C_1 = \left(g^{r_1},\, H((PK_1)^{r_1}) \oplus x_1\right)$$

Let's examine why this construction works:

1. For the chosen bit b:

$$\begin{aligned} C_b &= (V_1, V_2) \\ &= \left(g^{r_b},\, H((PK_b)^{r_b}) \oplus x_b\right) \\ &= \left(g^{r_b},\, H((g^k)^{r_b}) \oplus x_b\right) \\ &= \left(g^{r_b},\, H(g^{kr_b}) \oplus x_b\right) \end{aligned}$$

2. The receiver can compute:

$$\begin{aligned} V_1^k &= (g^{r_b})^k \\ &= g^{kr_b} \end{aligned}$$

Part 3: Decryption Process

The decryption succeeds because:

$$H\left((g^{r_b})^k\right) = H\left(g^{kr_b}\right) = H\left((PK_b)^{r_b}\right)$$

Therefore:

$$\begin{aligned} x_b &= \left(H\left(g^{kr_b}\right) \oplus x_b\right) \oplus H\left(g^{kr_b}\right) \\ &= x_b \oplus \underbrace{H\left(g^{kr_b}\right) \oplus H\left(g^{kr_b}\right)}_{=0} \\ &= x_b \end{aligned}$$

Part 4: Why No Information about x_{1-b} is Leaked

For the unchosen bit $(1-b)$, the receiver cannot decrypt C_{1-b} because:

- They don't know the discrete logarithm of PK_{1-b} with respect to g
- Due to the DDH assumption, $(PK_{1-b})^{r_{1-b}}$ appears random
- Therefore $H((PK_{1-b})^{r_{1-b}})$ acts as a one-time pad
- This makes x_{1-b} information-theoretically secure

Thus, we have shown that:

1. The receiver can successfully decrypt their chosen message x_b
2. The decryption process is efficient and deterministic
3. No information about x_{1-b} is revealed
4. The protocol achieves perfect correctness when followed honestly

□

4 Security Analysis

In this section, we provide an informal overview of security based on the standard assumptions. A fully rigorous proof would require reductions to the DDH or CDH problem in the random oracle model (depending on the exact variant and threat model).

4.1 Sender's Security

Goal: Show that a (possibly cheating) receiver B cannot learn both x_0 and x_1 with non-negligible probability under the DDH (or CDH) assumption.

Theorem 4.1: Sender's Security

Assume that the Decisional Diffie-Hellman (DDH) problem is hard in the group G. Then, in the Bellare-Micali 1-out-of-2 OT protocol, no polynomial-time receiver B can learn both messages x_0 and x_1 from the ciphertexts (C_0, C_1) with non-negligible probability.

Proof. We sketch a standard reduction-based argument. Suppose, for contradiction, that there exists a probabilistic polynomial-time (PPT) adversary B^* (a malicious receiver) that learns *both* x_0 and x_1 with non-negligible advantage. We show how to construct a PPT algorithm $\mathcal{D}$ that solves the DDH problem in G with non-negligible probability, contradicting the DDH assumption.

DDH Setup. Recall that the DDH problem in G is: given a random tuple (g, g^a, g^b, T), decide whether $T = g^{ab}$ or T is a random element in G. We assume $\mathcal{D}$ is given an input tuple (g, g^a, g^b, T) and must determine if $T = g^{ab}$. We will use this tuple to set up the OT protocol so that if B^* can learn both x_0 and x_1, then $\mathcal{D}$ can distinguish g^{ab} from a random element.

Simulation. The reduction $\mathcal{D}$ acts as the sender A in the OT protocol, interacting with B^*.

1. **Setup:** $\mathcal{D}$ chooses a random exponent $u \in \mathbb{Z}_p$ and sets

$$c = g^u \quad \text{(the published group element).}$$

   $\mathcal{D}$ sends c to B^*.

2. **Key Generation by B^*:** The adversarial B^* returns (PK_0, PK_1) such that $PK_0 \cdot PK_1 = c$ (or it may deviate arbitrarily if it tries to cheat).

3. **Encryption Phase:** The reduction $\mathcal{D}$ must produce the ciphertexts C_0 and C_1. Typically, A would choose $r_0, r_1 \in \mathbb{Z}_p$ uniformly at random and compute

$$C_0 = \Big(g^{r_0},\ H((PK_0)^{r_0}) \oplus x_0\Big), \quad C_1 = \Big(g^{r_1},\ H((PK_1)^{r_1}) \oplus x_1\Big).$$

   Here is where $\mathcal{D}$ embeds the DDH challenge into one of these components, ensuring that if B^* decrypts both, $\mathcal{D}$ learns something about T being g^{ab} or random.

   Concretely, $\mathcal{D}$ can, for instance, set:

$$g^{r_0} \leftarrow g^a, \quad g^{r_1} \leftarrow g^b, \quad \text{and} \quad H((PK_0)^{r_0}) \leftarrow H\big((PK_0)^a\big), \quad H((PK_1)^{r_1}) \leftarrow H\big((PK_1)^b\big).$$

   If $PK_0 = g^k$, then $(PK_0)^a = g^{ak}$; similarly, if $PK_1 = g^{u-k}$, then $(PK_1)^b = g^{b(u-k)}$, etc. The key point is that $\mathcal{D}$ can craft the XORed part to embed T (from the DDH challenge) in one of the two ciphertexts. For example, $\mathcal{D}$ might define

$$H((PK_0)^a) \oplus x_0 \ = \ \text{some function of } T, \quad H((PK_1)^b) \oplus x_1 \ = \ \text{some function of } T.$$

4. **Adversarial Decryption:** If B^* truly can obtain both x_0 and x_1, then B^* effectively learns enough information about $(PK_0)^a$ and $(PK_1)^b$ to distinguish whether $T = g^{ab}$ or a random element. Specifically, knowledge of both x_0 and x_1 reveals which hashed values correspond to valid Diffie-Hellman components.

Extracting the DDH Decision. If $T = g^{ab}$, then the embedded encryption values will yield consistent decryptions that B^* can exploit to recover both x_0 and x_1. If T is random, there is a non-negligible chance that B^*'s attempt to recover both messages will fail (or produce inconsistent values) because the underlying exponentiations will not match any legitimate Diffie-Hellman product. By observing B^*'s success or failure, $\mathcal{D}$ distinguishes between $T = g^{ab}$ and T random, thereby solving the DDH problem with non-negligible probability.

Conclusion. Hence, if B^* succeeds in learning both messages with non-negligible probability, then $\mathcal{D}$ breaks the DDH assumption. Since DDH is assumed to be hard in G, no such B^* can exist. Therefore, the Bellare-Micali OT protocol is secure against a malicious receiver trying to learn both messages. □

4.2 Receiver's Security

Goal: Show that the sender A cannot determine which bit $b \in \{0, 1\}$ is chosen by B.

Theorem 4.2: Receiver's Security

Under the Decisional Diffie-Hellman assumption, a sender A (even if malicious) cannot distinguish which of the two public keys PK_0 or PK_1 is associated with the receiver's actual choice bit b. Consequently, A cannot learn b with non-negligible advantage.

Proof. We again use a standard reduction argument. Suppose there exists a PPT sender A^* that can determine the choice bit b of the receiver B with non-negligible advantage. We construct a PPT algorithm $\mathcal{D}$ that breaks the DDH assumption.

DDH Setup. Let (g, g^x, g^y, T) be an instance of the DDH problem in G. The goal is to decide whether $T = g^{xy}$ or T is uniformly random in G.

Simulation as the Receiver. The algorithm $\mathcal{D}$ will simulate B's role and interact with A^* (the alleged distinguisher of b):

1. **Publishing c:** Typically, the sender A picks a random $u \in \mathbb{Z}_p$ and sends $c = g^u$ to B. In our simulation, $\mathcal{D}$ does nothing special here; it just obtains c from A^*. (Alternatively, $\mathcal{D}$ might control c if needed, but we can adapt how we embed the challenge.)

2. **Receiver Key Generation:** In the real protocol, the receiver chooses $k \in \mathbb{Z}_p$ and sets
$$PK_b = g^k, \quad PK_{1-b} = c/g^k.$$
The key point for security is that (PK_0, PK_1) is a random pair of group elements constrained by the product $PK_0 \cdot PK_1 = c$. Under the DDH assumption, to an outside observer (including A), (PK_0, PK_1) appears random (subject to the product constraint).
To embed the DDH challenge, $\mathcal{D}$ can decide which one of g^x or g^y stands for PK_0 or PK_1, effectively randomizing the assignment so that if A^* can guess b, it must be solving a DDH instance.

3. **Interaction and Challenge:** The malicious sender A^* attempts to determine b. But from A^*'s perspective, (PK_0, PK_1) is just some pair of elements in G such that $PK_0 \cdot PK_1 = c$. If $\mathcal{D}$ picks $PK_b = g^x$ and $PK_{1-b} = c/g^x$, the distribution of these values is indistinguishable from the real protocol if T is indeed g^{xy}. If T is random, we can adjust the scenario (for instance, one of these might not be a correct Diffie-Hellman share). In either case, any advantage A^* has in guessing b would yield an advantage in distinguishing $T = g^{xy}$ from T random.

Using A^*'s Advantage to Break DDH. If A^* truly can distinguish b with non-negligible probability, then by controlling how we assign (PK_0, PK_1) in terms of (g^x, g^y) and $c = g^u$, we can make A^*'s decision reveal information about whether $T = g^{xy}$ or random. Specifically:

- If T is a valid Diffie-Hellman term ($T = g^{xy}$), then the distribution of (PK_0, PK_1) is consistent with one fixed b.
- If T is random, then from A^*'s view, the assignment is inconsistent with a real Diffie-Hellman triple, changing the probability that A^* guesses b correctly.

Hence, any non-negligible bias in A^*'s guess of b translates into a non-negligible advantage in deciding whether $T = g^{xy}$, thus solving the DDH problem.

Conclusion. This contradiction implies that no polynomial-time sender A^* can guess the receiver's bit b with advantage better than $1/2 + \text{negl}(p)$. Hence, the protocol hides b from A, establishing receiver's security under the DDH assumption. □

4.3 Extensions and the Malicious Setting

A subtle point arises when the receiver is *malicious* (i.e., not following the protocol exactly). In such a scenario, the receiver might attempt to craft (PK_0, PK_1) in such a way to gain partial information about both x_0 and x_1.

Remark 4.3

When modeling H as a random oracle, one can strengthen the security argument to show that *any* adversarial strategy that breaks the OT properties would imply the ability to solve the Computational Diffie-Hellman (CDH) problem. For a more thorough treatment of malicious security, see [3].

5 Conclusion

The Bellare-Micali 1-out-of-2 Oblivious Transfer protocol demonstrates how classical public key techniques (like ElGamal) combined with hashing can yield a simple, elegant solution for oblivious transfer under the Decisional Diffie-Hellman assumption.

Key Points:

- The receiver's *choice bit* b is hidden due to the indistinguishability of PK_b vs. PK_{1-b} under DDH.
- The sender's messages x_0, x_1 remain protected, as the receiver can only decrypt one ciphertext properly.
- Extensions exist to handle more robust (e.g., malicious) adversaries in the random oracle model.

Open Problems / Future Directions:

- Adapting this protocol to post-quantum settings where DDH may be weaker.
- Extending to protocols that require *multiple* oblivious transfers or more complex secure computations.
- Practical optimizations, such as using elliptic curves or tailored hash functions for performance gains.

References

[1] Bellare, M., and Micali, S. "Oblivious Transfer and Polynomial Evaluation." *Advances in Cryptology—CRYPTO '92*, Springer, 1992.

[2] Diffie, W., and Hellman, M. "New Directions in Cryptography." *IEEE Transactions on Information Theory*, 1976.

[3] Naor, M., and Pinkas, B. "Oblivious Transfer and Polynomial Evaluation." *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC)*, 1999.

A Mathematical Details and Implementation Considerations

A.1 Properties of XOR Operation

The protocol's use of XOR ($\oplus$) is crucial for several reasons:

- **Perfect Secrecy:** When used with a random key of equal length, XOR provides information-theoretic security:
$$\forall m, c : \Pr[m \oplus k = c] = 2^{-n}$$
- **Computational Efficiency:** XOR operations are extremely fast in hardware:
  - Single CPU cycle per word-size operation
  - Parallelizable across multiple bits
  - No carry propagation needed
- **Self-Inverse Property:** XOR simplifies decryption because:
$$(m \oplus k) \oplus k = m$$

A.2 Group Selection and Security Implications

The choice of group G significantly impacts security and performance:

Group Type	Advantages	Disadvantages	Security Level
$\mathbb{Z}_p^*$ subgroups	- Well understood - Simple implementation	- Large parameters - Slower operations	3072-bit ≈ 128-bit security
Elliptic Curves	- Smaller parameters - Faster operations	- Complex implementation - Point validation needed	256-bit ≈ 128-bit security

Table 1: Comparison of Common Group Choices

A.3 Parameter Size Recommendations

Security levels should be chosen based on the intended application:

- **Classical Security Levels:**

$$\begin{aligned} &\text{80-bit:} && p \approx 1024 \text{ bits or 160-bit EC} \\ &\text{128-bit:} && p \approx 3072 \text{ bits or 256-bit EC} \\ &\text{256-bit:} && p \approx 15360 \text{ bits or 512-bit EC} \end{aligned}$$

- **Post-Quantum Considerations:** Parameters should be doubled for quantum resistance

A.4 Efficiency Analysis

Cost breakdown per operation:

- **Exponentiation:** $O(\log p)$ multiplications
- **Hash Function:** $O(1)$ for fixed input size
- **XOR Operation:** $O(n)$ bit operations for n-bit messages

Total computational complexity:

$$O(\log p) \text{ multiplications } + O(n) \text{ bit operations}$$

A.5 Implementation Considerations

Critical aspects for secure implementation:

1. **Random Number Generation:**
   - Use cryptographically secure RNG
   - Fresh randomness for each protocol run
   - Proper entropy collection
2. **Hash Function Requirements:**
   - Collision resistance: $2^{n/2}$ security
   - Pre-image resistance: 2^n security
   - Output length matching message size
3. **Group Element Validation:**
   - Verify elements are in the correct subgroup
   - Check order requirements
   - Reject invalid inputs

A.6 Performance Optimizations

Common techniques for improving efficiency:

- **Pre-computation:**
  - Fixed-base exponentiation tables
  - Window methods for exponentiation
  - Batch processing for multiple transfers
- **Implementation Tricks:**
  - Montgomery multiplication
  - Karatsuba multiplication for large numbers
  - Simultaneous multiple exponentiation

B Security Analysis Extensions

B.1 Malicious Adversary Considerations

Additional security measures for malicious adversaries:

- **Zero-Knowledge Proofs:**
  - Proof of knowledge for discrete logarithms
  - Proof of proper key generation
  - Range proofs for parameters
- **Protocol Augmentation:**
  - Commitment schemes
  - Cut-and-choose techniques
  - Verifiable random functions

B.2 Advanced Security Properties

Additional security features achievable with modifications:

- **Forward Secrecy:**
  - Ephemeral keys
  - Session key derivation
  - Perfect forward secrecy options
- **Fairness:**
  - Gradual release mechanisms
  - Timeout-based aborts
  - Trusted setup options