On Necessary and Sufficient Conditions for Private Ballot Submission

D. Bernhard[1], O. Pereira[2], and B. Warinschi[1]

[1] University of Bristol, England
[2] Université catholique de Louvain, Belgium

Abstract. We exhibit the precise security guarantees that a public key encryption scheme needs to satisfy to guarantee ballot privacy when used in a large class of voting systems. We also identify new security notions for public key encryption that characterize the number of times that a public key can be used in different elections, and show that the most common ballot preparation approach that consists in encrypting the vote and adding a NIZK proof of its validity is sound, even without hardwiring the voter identity in the proof. Our results provide important steps towards proving the privacy of the ballot submission procedure in the widely deployed Helios voting system.

1 Introduction

Public key encryption is a central primitive used to guarantee the privacy of the votes in many classical voting schemes [7, 20, 11, 13]. Since numerous standard security notions exist for public key encryption [3], it is quite natural to wonder what security level is appropriate for ensuring ballot privacy.

This question has been addressed in several previous works, which show that schemes satisfying the strongest usual security notion for public key encryption, namely IND-CCA2 security, are suitable for ballot preparation: Bernhard et al. [9] show this with respect to a game-based notion of ballot privacy, while Wikström [22] shows this with respect to a notion of secure input submission to functionalities in the UC framework. The intuition underlying these works is that IND-CCA2 security guarantees confidentiality of the submitted votes, but also independence of the votes by preventing any voter from submitting a vote that is a transformation of someone else's vote; mere ballot copies can easily be rejected by the voting server.

However, the voting schemes referenced above, as well as the one implemented in the Helios voting system [2] which is now widely deployed, do not use an encryption scheme that is known to be IND-CCA2 secure under any "classical" assumption: for Helios, for instance, one would expect a security proof in the random oracle model relying on the hardness of the DDH problem, but building such a proof is a long-standing open problem in cryptography [21].

As a result, Bernhard et al. [9] recently suggested modifying the ballot preparation protocol of Helios in order to strengthen the encryption mechanism that is used there (using the Naor-Yung transform [19, 15] on ElGamal), at the cost of substantially increasing the computational load of ballot preparation. This change in the encryption scheme is however not motivated by any attack on the ballot preparation protocol used in Helios. Since the efficiency of the ballot preparation procedure can be a serious limiting factor when the number of election candidates increases [17], it is of central importance to actually identify the *minimal* conditions that an encryption scheme must satisfy to ensure ballot privacy. This would allow one to determine whether the proposed augmentation of the computational load for ballot preparation is actually necessary.

Our contributions. Our contributions to this question are as follows.

1. We reduce the demand on the security of encryption from IND-CCA2 to NM-CPA. For the latter notion we show that it is both necessary and sufficient to guarantee ballot privacy in a single election.
2. Considering the possibility of reusing a public key in multiple elections, as proposed in Helios for instance in order to simplify the key management, we define a new family of security notions for public key encryption which we term IND-k-CPA. This is precisely the level of security needed to allow reusing a key in k elections. The IND-k-CPA security game is identical to the IND-CPA game excepted that the adversary is allowed to perform k decryption queries after the test query, each containing an arbitrary number of ciphertexts. In particular NM-CPA is equivalent to IND-1-CPA.
3. We prove the soundness of the common ballot preparation approach that consists in encrypting a vote using an IND-CPA encryption scheme and adding a NIZK proof of the validity of the ballot, provided that the ZK proof satisfies some (commonly satisfied) conditions. In contrast to other existing proposals our proof is valid even if the identity of the voters is not hardwired in the ballots.
4. Finally, we suggest a modification of the ballot preparation protocol in the Helios voting system, which provides provable ballot privacy at a negligible computational cost (compared to a 50% cost increase in previous work) and without changing the size of the ballots (compared to a a 50% cost increase in previous work as well).

Related work. We consider related work in terms of voting protocol analysis and of notions of public key encryption.

Voting protocol analysis. In one of the first proofs of the security of voting schemes, Groth [16] analyzes in the UC framework protocols that are similar to those we consider. In those protocols, ballot independence is obtained in a way that differs from the one we consider: the voter ID is included in the random oracle input provided when computing all the ballot validity proofs made non-interactive using the Fiat-Shamir heuristic.

Benaloh [8] however indicates how substantial benefits can be gained in terms of verifiability of the ballot preparation device when that device does not know the identity of the person who uses it: ballot preparation can be separated from ballot submission, during which identification takes place only after the ballot preparation device has committed to the content of the ballot to be submitted. This is the procedure that is adopted in Helios, and takes the protocol used therein out of the scope of Groth's proof.

We, on the other hand, investigate the type of non-malleability that needs to be guaranteed by the encryption scheme in order to make it possible to enforce ballot independence just by rejecting ballot copies. The two techniques are very different in spirit:

– Relying on IDs included in the proofs requires schemes that make it infeasible to manipulate values hardwired inside proofs. As a result, this approach does not preclude using proofs that would be completely re-randomizable: ballot independence is verified by checking the validity of proofs with respect to the identity of their sender.
– Relying on non-malleability, on the other hand, requires the use of schemes that cannot be re-randomized, since independence is now enforced by rejecting duplicate ciphertexts. In this case, ballot independence is enforced without checking the validity of any proof.

More recently, Küsters et al. [18] proposed a notion of vote privacy that is particularly well tailored for non cryptographic schemes, as well as for measuring the privacy guaranteed by a voting system, i.e., the privacy that voters keep on their vote once the tally is revealed. The notion of Bernhard et al. that we use is more tailored to measure the quality of a ballot submission procedure, independently of the tallying procedure that is adopted.

Security notions for public key encryption. Our notion of IND-k-CPA security is closely related in appearance to the notion of q-bounded CCA security of Cramer et al. [12]: the q-bounded CCA security game is equivalent to the IND-CPA game, except that the adversary is allowed to perform q decryption queries, each for one ciphertext. The resulting hierarchy of security notions is however located in a very different space than the one produced by IND-k-CPA: while IND-0-CPA and 0-bounded CCA security are equivalent to IND-CPA security, Cramer et al. show that for every value of q, the q-bounded security notion is strictly weaker than NM-CPA security. On the other hand, NM-CPA security is equivalent to our IND-1-CPA security notion [6], and IND-CCA2 security implies IND-k-CPA security for every k.

2 Preliminaries

In this section we introduce some of the notions that we use throughout the paper and fix notation.

A cryptographic scheme or protocol is formally a map taking a security parameter $\lambda \in \mathbb{N}$ to a set of algorithms indexed by λ. For example, a public-key encryption scheme for a given security parameter λ is a triple $(\mathsf{Gen}_\lambda, \mathsf{Enc}_\lambda, \mathsf{Dec}_\lambda)$. We follow the common convention that security parameters are not written out in algorithm names and will thus write a public-key encryption scheme simply as $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The "setup" algorithm of a scheme will take the security parameter as input, encoded in unary; we write this as 1^λ.

The security notions that we use in this paper are mostly based on indistinguishability games. Here we give a definition that is sufficiently abstract to apply to multiple scenarios of interest, in particular to privacy of plaintexts for encryption and privacy of ballots in voting schemes. An indistinguishability game consists of a probabilistic process C, known as a challenger, that provides a well-defined interface for an adversary A. Both the challenger and the adversary receive the security parameter λ in unary encoding 1^λ as input.

An indistinguishability game begins by having the challenger select a bit β uniformly at random; its later execution may depend on this bit. An adversary may at any time output a bit α, in which case the challenger halts. We write $\alpha_{A,C}(\lambda)$ for the random variable denoting the bit α output by adversary A after interacting with challenger C when 1^λ was given to both as input. We define the *advantage* of A against C as the quantity

$$\mathbf{Adv}_{A,C}(\lambda) := |\mathbf{Pr}[\alpha_{A,C}(\lambda) = 1 \mid \beta = 1] - \mathbf{Pr}[\alpha_{A,C}(\lambda) = 1 \mid \beta = 0]|$$

The advantage of an adversary ranges from 0 (if the adversary's output is uniform and independent of β) to 1 (for example if $\alpha = \beta$ with probability 1).

An *efficient* adversary is one whose running time is bounded by a polynomial in λ. A function $\nu : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is *negligible* if for any $c \in \mathbb{N}$ we have that $\lim_{\lambda \to \infty} \nu(\lambda) \cdot \lambda^c = 0$.

We call a scheme (computationally) secure with respect to a game C if for any efficient adversary A, $\mathbf{Adv}_{A,C}(\lambda)$ is a negligible function of λ.

Public Key Encryption. In this section we recall the notion of public key encryption, recall some of the security notions in use, and discuss some issues relevant to the goal of this paper.

Definition 2.1 (Public Key Encryption). *A public-key encryption scheme is a triple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfying the following description and correctness condition.*

$\mathsf{Gen}(1^\lambda) \mapsto (pk, sk)$ The key generation algorithm takes a security parameter and outputs a pair (pk, sk) of items called the public key and secret key.

$\mathsf{Enc}(pk, m) \mapsto c$ The encryption algorithm takes a public key pk and a message m and outputs a ciphertext c.

$\mathsf{Dec}(sk, c) \mapsto m$ The decryption algorithm takes a secret key sk and a ciphertext c and returns a message m or the special symbol $\perp$ to indicate an invalid ciphertext.

A public-key encryption scheme must satisfy the following *correctness* property. For any message m and any security parameter λ, the procedure

$$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda); c \leftarrow \mathsf{Enc}(pk, m); m' \leftarrow \mathsf{Dec}(sk, c)$$

will yield $m' = m$.

Security notions for asymmetric encryption. Of the many security notions developed for asymmetric encryption we are mainly interested in two: indistinguishability under chosen-plaintext attack (IND-CPA) and non-malleability under chosen-plaintext attack (NM-CPA). Non-malleability is difficult to work with, so we use an equivalent formulation called IND-1-CPA security. The difference between this notion and IND-CPA security is that in addition to standard IND-CPA abilities, the adversary is allowed *one* parallel decryption query (where he outputs a vector of ciphertexts and obtains the underlying plaintexts). More generally, we consider IND-k-CPA security where the adversary is allowed at most k (parallel) decryption queries. We recall these notions in Appendix A and discuss implications and separations between them. In particular, we show that IND-$(k+1)$-CPA encryption is a strictly stronger notion that IND-k-CPA.

A result on which we rely in this paper is the following relation between IND-1-CPA and NM-CPA, due to Bellare and Sahai [6].

Proposition 2.2 (NM-χ = IND-1-χ). *For any attack scenario*

$$\chi \in \{\mathit{CPA}, \mathit{CCA1}, \mathit{CCA2}\}$$

the notions IND-1-χ and NM-χ of public-key encryption security are equivalent.

Sigma protocols. We also recall the notion of Sigma protocols: we use these to construct non-malleable encryption.

Definition 2.3 (Sigma Protocol). *A Sigma protocol for an NP language*

$$\mathcal{L} : \{s \mid \exists w : L(s, w) = 1\}$$

is a pair of interactive algorithms (P, V) that work as follows:

Prover The algorithm for the prover takes as input a statement s and a witness w.

1. First, P outputs a "commitment" *comm* .
2. It waits for a challenge *chal* .
3. P sends a "response" *resp* and halts.

A state is passed from stage one to stage 3.

Verifier The protocol for the verifier takes as input a statement s.

1. The protocol V receives commitment *comm* .
2. It selects a challenge *chal* uniformly at random from a challenge space and sends it to the prover.

3. Upon receiving a response $resp$ on the prover interface the verifier evaluates a predicate Verify on the statement s and the transcript $(comm, chal, resp)$ and returns 0 or 1, then halts.

In addition to completeness (an honest run between $P(s, w)$ and $V(s)$ always accepts if $L(s, w) = 1$) we also require special soundness and special honest verifier zero knowledge.

Definition 2.4 (Special Soundness). *A* matching pair *of transcripts with respect to a statement s is a pair of transcripts $t_1 = (comm_1, chal_1, resp_1)$ and $t_2 = (comm_2, chal_2, resp_2)$ such that both transcripts verify (w.r.t. p and s), the commitments are equal $(comm_1 = comm_2)$ but the challenges are different $(chal_1 \neq chal_2)$.*

A sigma protocol has special soundness if there is an extractor $\mathsf{Extract}$ *that takes as input a statement s and a matching pair of transcripts (t_1, t_2) and returns a witness w such that $L(s, w) = 1$.*

Definition 2.5 (Special Honest Verifier Zero Knowledge). *A sigma protocol has* special honest verifier zero knowledge *if there is a simulator* Sim *that takes as input a statement s (that may or may not be valid) and a challenge $chal$ and outputs a transcript $(comm, chal, resp)$ using the challenge provided such that* $\mathsf{Verify}(s, (comm, chal, resp)) = 1$. *Furthermore, transcripts produced by the simulator for correct statements s are indistinguishable from transcripts produced by P and V on input s, where P additionally has any witness w for s as input.*

Finally, we are interested in sigma protocols that have the *unique response* property. This property is not assumed implicitly for any sigma protocol and is as follows: for any statement, commitment and challenge, there exists at most one response that would lead to an accepting transcript.

Definition 2.6 (Unique Responses). *A sigma protocol has* unique responses *if for any statement s, any commitment $comm$ and any challenge $chal$ there is at most one value $resp$ such that* $\mathsf{Verify}(s, (comm, chal, resp)) = 1$.

3 Single-Pass Voting Schemes

In this section we discuss the security of an important class of voting schemes called single-pass [9]. This class includes Helios [1] which is the focus of this paper. These are protocols for parties including a set $\mathcal{V}$ of voters, a set $\mathcal{A}$ of administrators and a bulletin board BB. The execution of such protocols takes place in three phases: *setup*, *voting* and *tallying*.

More formally, a single-pass scheme is described by the following six algorithms and an associated execution protocol. The first three represent the actions of the principal actors in the different phases of the protocol:

$\mathsf{Setup}(1^\lambda)$ is an interactive protocol for the administrators. It takes a security parameter λ as input and produces a single public output and a private output for each administrator. We write $(x, y) \leftarrow \mathsf{Setup}(1^\lambda)$ for this process (where y is the public output).

$\mathsf{Vote}(v, y)$ is a probabilistic algorithm run by voters to prepare their votes for submission. It takes as input vote v and public information y and outputs a ballot $s \leftarrow \mathsf{Vote}(v, y)$.

$\mathsf{ProcessBallot}(b, s)$ runs during the voting phase, takes the bulletin board b and a new submission s as inputs and checks the submission for validity. The validity check may depend on the board b as well as the actual submission s (this addresses the issue of repeated submissions). Upon success the algorithm returns a new state for the board b' and it also informs the caller on the success of the operation. We write $(r, b') \leftarrow \mathsf{ProcessBallot}(b, s)$ for this process, and assume that $r = \perp$ in the case the operation did not succeed.

$\mathsf{Tally}(x, b)$ is an interactive protocol for the administrators. The algorithm takes the private input of the administrators, the current state of the board and outputs a result $r \leftarrow \mathsf{Tally}(x, b)$.

A voting scheme also comes with verification algorithms. Since we are only concerned with privacy, with omit to specify these.

A single-pass scheme is executed as follows.

1. *Setup phase.* The administrators jointly run the Setup subprotocol and post its public output m to the bulletin board.
2. *Voting phase.* Each voter may proceed as follows or abstain. The voter produces its ballot $s \leftarrow \mathsf{Vote}(v, y)$ where v is his input vote and y is the public key associated to the election. The voter then submits s to the board who then runs the $(r, b') \leftarrow \mathsf{ProcessBallot}(b, s)$ where b is the current state of the board, and updates the state of the board to b'.
3. *Tallying phase.* The administrators first simulate the board's actions in the voting phase (e.g. in particular checking that the board never added any data that failed the $\mathsf{ProcessBallot}$ test). Next, they jointly run Tally and append the result to the board.

We assume that there is a well-defined way to identify the messages sent in the different phases on a bulletin board.

A "minivoting" scheme. An example of a single-pass scheme is the generic construction called "minivoting"[9]. The scheme is constructed generically from any public-key encryption scheme. It requires a trusted administrator and authentic but public channels. The resulting scheme offers privacy, but not verifiability and serves a dual purpose. On the one hand it demonstrate the role of encryption in ensuring ballot privacy. On the other hand, it allows a modular proof of Helios where one first shows the privacy of a related minivoting scheme, and then one argues that the additional changes that takes minivoting to Helios does not affect the privacy property.

The construction works for any result function ρ and is as follows. Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Then the induced *minivoting* scheme $\mathsf{Enc2Vote}(\Pi)$ for the result function ρ is the following single-pass scheme.

$\mathsf{Setup}(1^\lambda)$ Run $\mathsf{Gen}(1^\lambda)$ to produce a key pair (pk, sk); the public output is pk and the secret one, sk.

$\mathsf{Vote}(v, y)$ Encrypt v with $y = pk$: $c \leftarrow \mathsf{Enc}(pk, v)$. Return c.

$\mathsf{ProcessBallot}(b, s)$ If the new submission s already appears on the board b, reject it; otherwise accept it and add it to b.

Tally Decrypt all ballots $\mathbf{b}$ on the board using sk (this can be done in parallel) to get the underlying votes $\mathbf{v}$ and evaluate $r \leftarrow \rho(\mathbf{v})$; return r. (Here we use $\mathbf{b}$ and $\mathbf{v}$ to represent all of the ballots and underlying votes, respectively).

Ballot Privacy. In this paper we use (an equivalent variant) of the recently proposed notion of ballot privacy for single-pass voting schemes [9]. Specifically we use a model where privacy is captured via a left-or-right game, whereas in the original definition privacy is modeled as a real-or-random game.

Informally, the model of [9] considers an adversary that controls a set of dishonest voters on whose behalf he can submit arbitrary ballots. In addition, the adversary chooses votes for the honest voters and observe their ballots; these choices can be made adaptively. At the end of the election the adversary obtains the election result. The adversary has to distinguish between a world where he observes the real execution (as outlined above) or an execution where the ballots of honest parties are replaced with ballots for some fixed vote.

In the version that we consider in this paper, instead of submitting a single vote for honest parties, the adversary submits two votes. The vote to be cast is selected according to an internal uniformly random bit, which the adversary aims to guess. It is immediate (via a simple hybrid argument) that the two security notions are equivalent.

Definition 3.1 (Ballot Privacy). *A single-pass protocol has* ballot privacy *if for any efficient adversary A, the advantage $\mathbf{Adv}_{A,C}(\lambda)$ against the following indistinguishability game with challenger C is negligible as a function of λ.*

Setup phase. The challenger picks a bit $\beta \overset{R}{\leftarrow} \{0,1\}$ uniformly at random. He sets up two empty bulletin boards L and R runs the Setup protocol on behalf of the administrators to obtain (x, y) and posts the public information y on both boards. The adversary is given access to either L if $\beta = 0$ or to R if $\beta = 1$.

Voting phase. The adversary may make two types of queries.

$\mathsf{Vote}(id, v_L, v_R)$ queries. The adversary provides a voter identity id and two votes (v_L, v_R). The challenger runs $b_L \leftarrow \mathsf{Vote}(v_L, y)$ and $b_R \leftarrow \mathsf{Vote}(v_R, pub)$, where pub is the public information on the boards from the setup phase. He then obtains new versions of the boards L and R by running $\mathsf{ProcessBallot}(L, v_L)$ and $\mathsf{ProcessBallot}(R, v_R)$, respectively.

$\mathsf{Ballot}(b)$ queries. These are queries made on behalf of corrupt parties. Here the adversary provides a ballot b. The challenger runs $\mathsf{ProcessBallot}(L, b)$, and if the process accepts it also runs $\mathsf{ProcessBallot}(R, b)$.

Tallying phase. The challenger evaluates the L board and posts the result on *both* boards.

Remark 3.2. Notice that the final result that is placed on the boards is the result of the "left" voting process. This is necessary in order to prevent the adversary from distinguishing between the two boards in a trivial way (by submitting votes that lead to different results). A weaker, but nonetheless interesting security definition would allow the adversary to see the actual results both on the left and on the right board, provided that the choices of honest votes are such that they would lead to the same result (e.g. they are permutations of one another in a typical election).

4 Ballot Privacy is equivalent to Non-Malleable Encryption

As explained in the introduction, IND-CCA2 seems to be necessary for proving the security of a voting scheme: the adversary submits ballots on behalf of the dishonest voters and it may seem (from the description of the ballot privacy game) that the challenger need to decrypt these ballots in order to provide the correct results of the election. Indeed, the previous paper that looks at the security of Helios [9] argues its security if the underlying encryption scheme is modified to offer CCA2 security. More precisely, that paper argues that the minivoting constructed from the underlying encryption scheme (which is then essentially a stripped down version of Helios) is secure if the encryption scheme is IND-CCA2:

Theorem 4.1 (IND-CCA2 is sufficient for ballot privacy). *Let Π be an IND-CCA2 secure encryption scheme. Then* $\mathsf{Enc2Vote}(\Pi)$ *has ballot privacy.*

In this paper we show that the intuition which lead to the requirement that the underlying encryption scheme be IND-CCA2 is not accurate. More precisely, we show that the weaker notion of non-malleable CPA encryption (or equivalently that of IND-1-CCA) suffices. We prove that:

Theorem 4.2 (NM-CPA is sufficient for ballot privacy). *Let Π be an NM-CPA secure encryption scheme. Then* $\mathsf{Enc2Vote}(\Pi)$ *has ballot privacy.*

Proof (Theorem 4.2). Recall that the IND-1-CPA game allows the adversary one challenge query and one parallel decryption query with an arbitrary number of ciphertexts but the adversary must ask the decryption query after the challenge query.

We prove the theorem by reduction: if there is an efficient adversary A against the minivoting scheme that makes at most q queries (vote or ballot) and has advantage α then we construct an efficient adversary A' against the IND-1-CPA game with advantage at least $\alpha/(2q)$.

Let A be an adversary against $\mathsf{Enc2Vote}(\Pi)$ and let q be a bound on the number of queries (vote and ballot) that A can make. (Both A and q may depend on λ; the efficiency of A guarantees that one can choose a q that is polynomial in λ.) We consider a sequence of $q + 1$ hybrid games labeled H_0 up to H_q. In each hybrid, we also keep track of a special third board (this board serves the purpose of tallying the "real" election which is that given by the left votes). We refer to this third board as the middle board. The hybrid games are defined as follows. In game H_i, for the first i vote queries a ballot for the left vote v^L is placed on the visible board whereas all queries from the $i + 1$-st the right vote v^R is used. In all these hybrid games, whenever the adversary makes a ballot query (i.e submits a ballot on behalf of a dishonest party) the ballot is placed on the middle board first, then on both others if the duplicate-check on the middle board did not reject it. The tally is computed always using the middle board.

Notice that per the above description, the view of the adversary against H_0 is exactly the same as in the ballot privacy game in the case when the secret bit β is 0 and in H_q he sees exactly the same distribution as in the ballot privacy game when $\beta = 1$. It is clear that if an adversary A can distinguish H_0 from H_q with advantage α then there is a pair (H_i, H_{i+1}) that he can distinguish with advantage at least α/q.

We now show how to construct an adversary against the IND-1-CPA security of the underlying encryption scheme.

- We obtain the election public key from the encryption challenger.
- We use our one challenge query for the single query in which the two hybrids in question differ in their behavior: we submit the pair (v^L, v^R) as our two messages and post the challenge ciphertext on the board as the resulting ballot.
- To tally, we use our one parallel decryption query to decrypt all the ballots submitted by dishonest users and combine the decrypted votes with all the "left" votes of honest users to obtain the result.

  Notice that the resulting adversary is a valid one: the challenge ciphertext is not submitted to the decryption query as the board submission protocol prevents duplicate ballots.
- We forward A's guess at the challenge bit to our challenger.

In the reduction using the pair (H_i, H_{i+1}) (for any $i \in \{0, \ldots, q - 1\}$), if the encryption challenger picked $\beta = 0$ then the adversary A sees the distribution of H_i and if $\beta = 1$, that of H_{i+1}. Therefore, we can simply take A's guess at the challenge bit and forward it to our challenger to win with the same advantage.

It remains to compute the advantage of our reduction. The transition from NM-CPA to IND-1-CPA loses at most a factor 2 by Proposition 2.2 and the hybrid argument at most a factor q. If A had advantage α, this gives $\alpha/(2q)$ as claimed. $\square$

One interpretation of the above result is that, intuitively, to ensure ballot privacy it is sufficient to ensure that it is impossible to cast ballots such that the underlying vote is related in a meaningful way to those ballots that have been already cast. It is natural to ask if this condition is also necessary. Next we show that indeed this is the case in a sense that we now make more precise.

The idea of the construction is to show that the ballot-casting algorithm is in some sense equivalent to a non-malleable encryption algorithm. The condition we require is that we can build a parallel decryption algorithm from the tallying procedure. This holds in the special case that the result function is the identity function, i.e. while voters have privacy during the ballot-casting phase all votes are revealed at the end of the election. This result shows that non-malleable encryption is not only sufficient but in a strong sense also necessary to construct a voting scheme with ballot privacy.

For a voting scheme $\Sigma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{ProcessBallot}, \mathsf{Tally})$ we define the encryption scheme $\mathsf{Vote2Enc}(\Sigma)$ as follows. The key generation algorithm Gen simply runs the Setup algorithm of Σ to obtain (x, y) where y is the public output of Setup and x is the set of private outputs. The plaintext space of the resulting encryption scheme is the same as the space of valid votes. The encryption of message m under the public key y runs the ballot-preparation algorithm for some arbitrary but fixed identity to obtain a ballot $b \leftarrow \mathsf{Vote}(m, y)$. Finally, to decrypt a ciphertext b, one sets up a board B, casts b (i.e. runs $\mathsf{ProcessBallot}(B, b)$ to obtain a new board B and runs the tallying procedure. Since the board is freshly created, there are no other votes cast on it, so this process is equivalent to one that tallies a board with only ballot b on it, i.e. it returns $\mathsf{Tally}(x, b)$.

Correctness of the resulting encryption scheme follows from the assumption that the tallying function does indeed return the set of cast votes (in this case precisely the plaintext encrypted in the cast ballot). Next, we show that if Σ has ballot privacy then the resulting encryption scheme is NM-CPA.

Theorem 4.3. *Let Σ be a voting scheme for the result function ρ that returns all votes in the order cast. If Σ has ballot privacy then* $\mathsf{Vote2Enc}(\Sigma)$ *is an NM-CPA encryption scheme.*

Proof. We reduce IND-1-CPA of the encryption scheme to ballot privacy of the voting scheme. Let A be an adversary against the encryption scheme.

- We start by obtaining the public key from the ballot-privacy challenger and forwarding it to A.
- When A makes his challenge query with two messages (m_0, m_1) we make a vote query with these two messages and return the resulting ballot as the challenge ciphertext.
- When A makes his parallel decryption query, we post all his ciphertexts to the board using ballot queries and ask for the election to be tallied, giving us the plaintexts which we return to A.

  The parallel decryption query may not contain the challenge ciphertext so we will not run into problems with the duplicate-checking algorithm.

  Note that it is important that the decryption query can only be asked after the challenge query.
- When A outputs a bit we forward this to our challenger.

The distribution that A sees will be exactly that of the IND-1-CPA game, the secret bit β corresponding to the one chosen by our challenger, so we win the ballot privacy game with the same advantage as the adversary has against the encryption scheme. □

5 Encrypt + PoK

In this section we study one approach towards constructing non-malleable encryption schemes, namely appending to the ciphertext of some IND-CPA scheme a proof of knowledge of the plaintext. More precisely we look at the case when the proof of knowledge is derived from a Sigma-protocol for proving knowledge of a plaintext underlying a certain ciphertext.

Theorem 5.1 (NM-CPA of Enc+FS-PoK). *Let $\mathfrak{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure encryption scheme and $\mathfrak{P} = (P, V)$ be a sigma protocol for the language*

$$L((pk, c), (m, r)) = 1 \iff c = \mathsf{Enc}(pk, \underline{m}; \underline{r})$$

with special soundness, special honest verifier zero knowledge and unique responses and a large challenge space C (i.e. $1/|C|$ is negligible). Let H be a random oracle $\{0,1\}^* \to C$ and let $\|$ denote an unambiguous and efficient encoding $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$.

Then the following construction $\mathfrak{E}'$, which we call the Encrypt+FS-PoK construction*, is an NM-CPA secure encryption scheme.*

Gen' Run Gen to create (pk, sk) and return these.

Enc' Given pk and a message m, pick a random value r and run $c \leftarrow \mathsf{Enc}(pk, m; r)$ to create an "original" ciphertext c; run P on input $((pk, c), (m, r))$ until he creates a commitment $comm$; compute $chal \leftarrow H(pk\|c\|comm)$ and send this to P; obtain the response $resp$ from P and return the ciphertext $c' \leftarrow (c, comm, resp)$.

Dec' Parse the ciphertext c' as $(c, comm, resp)$. If $\mathsf{Verify}((pk, c), comm, chal, resp)$ is false then return $\perp$. [3] Else, decrypt c with Dec and sk and return the plaintext.

Proof. We show IND-1-CPA security of the construction $\mathfrak{E}'$ which is equivalent to NM-CPA by Proposition 2.2.

Let C be a challenger for the IND-CPA game of $\mathfrak{E}$. Let A be an adversary against the IND-1-CPA property of $\mathfrak{E}'$. We construct an "adapter" B that uses A to attack C.

- B begins by obtaining a public key pk from C and forwarding it to A.
- B simulates the random oracle towards A and keeps a list of all previously asked queries. Whenever A queries a value already on the list, B answers consistently. Whenever A queries a new value, B chooses a response uniformly at random from the oracle's range, adds an entry to the list and returns the response. In addition, B will occasionally add entries of its own to the list; this is known as *patching the oracle.*
- When A makes a challenge query with two messages (m_0, m_1), B forwards these to C to get a challenge ciphertext c^*. B then picks a challenge $chal^*$ uniformly at random from the challenge space and runs the sigma protocol's simulator that exists by the "special honest verifier zero knowledge" property on this challenge to obtain a simulated transcript $\pi^* = (comm^*, resp^*)$.
  B patches the oracle (list) such that $H((pk, c^*)\|comm^*) = chal^*$. If this fails (because the preimage is already on the list), B aborts with "Error 1".
  B can now return (c^*, π^*) to A.
- When A makes his decryption query with a vector $\mathbf{c}$ of ciphertexts, B acts as follows. For each index i into the vector $\mathbf{c}$, B executes the following process.
  B first checks if ciphertext c_i is valid, that is it is of the form $(c_i^0, \pi_i = (comm_i, resp_i))$ where $\mathsf{Verify}((pk, c_i^0), (comm_i, H((pk, c_i^0)\|comm_i), resp_i)) = 1$; if not then this component "decrypts" to $\perp$. Similarly if A is trying to decrypt the challenge ciphertext (c^*, π^*) he gets $\perp$ in return. If the point $(pk, c_i^0)\|comm_i$ is not on the oracle list yet, B adds it as if A had asked the query while verifying the proof.
  B next finds the entry on his oracle list where A asked $(pk, c_i^0)\|comm_i$ to the oracle; if no such entry exists or the entry was made by B himself (while patching the oracle) then B

[3] We assume w.l.o.g. that the decryption algorithm has access to the public key pk as well as the secret key sk.

aborts with "Error 2". B simulates a fresh copy of A identically up to the point where the above query is asked for the first time and answers this query with a different, uniformly random value. He continues this simulation until A outputs his vector $\mathbf{c}'$. If $\mathbf{c}'$ contains an entry $(c_i^{0\prime}, (comm_i', resp_i'))$ that is a valid ciphertext in the simulated run and with $c_i^{0\prime} = c_i^0$ and $comm_i' = comm_i$ then B runs the special soundness extractor to get the witness w_i for this statement. Note that a valid proof in the simulated run of A will by construction induce a different challenge than in the "main" run.
If the simulation does not produce the desired results, B simply repeats it until it does.
– When A outputs his guess at the challenge bit, B forwards this to C and halts.

The *forking lemma* of Bellare and Neven [5] guarantees that for each value of i, B will find a witness in expected polynomial time. As the number of entries in $\mathbf{c}$ is itself polynomial in the security parameter, B therefore finds *all* witnesses in expected polynomial time.

Suppose that A had non-negligible advantage against the IND-1-CPA game of $\mathfrak{E}'$. As long as B does not abort, the only difference A sees (in the "main" run) is that the proof π^* he gets in response to his challenge query had been produced using the simulator. A cannot detect this change except with negligible probability (or otherwise A breaks the zero-knowledge property of the Sigma protocol.

It remains to argue that B only aborts with negligible probability. "Error 1" occurs if A has already queried the oracle at $(pk, c^*)\|comm^*$ where c^* is the challenge ciphertext — before A has produced this ciphertext! An adversary that can predict ciphertexts of messages with non-negligible probability immediately yields one against IND-CPA security of the encryption scheme.

"Error 2" occurs in one of two cases. The first is that A outputs a valid ciphertext containing a proof with a challenge (the oracle output on $(pk, c)\|comm$) that is not on the oracle list at all. However, to check validity of a proof B algorithm must call the oracle at this point too; because the point is not on the list yet a new value will be chosen uniformly at random and hence the probability of A having guessed the correct one out of a large challenge space is negligible.

The other case that could raise this error 2 is that the relevant entry is on the hash list but B put it there while patching the oracle to deal with A's challenge query. In this case, for the relevant index i, $(pk, c_i^0)\|comm_i = (pk, c^*)\|comm^*$. As the encoding $\|$ is unambiguous we get that $c_i^0 = c^*$ and $comm_i = comm^*$. Because the relevant proof must be valid (or else B would already have returned $\perp$) we also have that $chal_i = chal^*$. But our sigma-protocol has unique responses so $resp_i = resp^*$ as well. In other words, A is asking to decrypt the challenge ciphertext but we would already have returned $\perp$ in this case too. This is a contradiction so this case cannot happen.

We conclude that B only aborts with negligible probability and as long as B does not abort, A cannot tell this reduction from the real IND-1-CPA game. Therefore, if the advantage of A was non-negligible then so is that of B against C. □

6 Ballot privacy in Helios

The results described above substantially close the gap towards proving the security of the ballot submission procedure used in the homomorphic tallying based version of Helios [2] (the mixnet-based version [10] already uses a provably secure submission mechanism).

Bernhard et al. [9] suggested replacing the ElGamal encryption of each vote with an IND-CCA2 encryption of the same vote based on the Naor-Yung transform. A straightforward instance of this transform based on a sigma proof makes the complexity of the encryption move from two exponentiations to 8. A simple reorganization of the proof proposed by Fouque and

Pointcheval reduces this to 7 exponentiations. Bernhard et al. also point another trick, proposed by Bellare et al. [4], that allows reducing this to 6 exponentiations. The variant of the TDH2 scheme proposed by Bulens et al. [10] requires 5 exponentiations. If we add the complexity of the 0/1 disjunctive proof to this, which takes 4 exponentiations, we then have a ballot complexity preparation of 9 exponentiations compared to the 6 that appear in the current version of Helios (for approval voting).

The modifications that are needed in the ballot preparation protocol used in Helios in order to be able to apply our proofs are much more benign: we only need to change the input of the hash function that is used in the 0/1 disjunctive proofs and to add the ciphertexts there, together with the election id which is unique to each election. The corresponding overhead is negligible compared to the extra modular exponentiations that were required before, and does not change the size of the ballots at all, compared to the other solutions which required to send at least 3 extra group elements.

Reusing keys. Can one safely reuse a key pair for several elections without breaking ballot privacy? (Note that this is a purely theoretical question. There are many practical reasons why one would discourage such reuse of keys.)

Our security proof for minivoting schemes is by reduction to IND-1-CPA: the one parallel decryption query is used to recover votes produced by the adversary. In essence, this already shows that one can run a single poll with multiple questions, each question being tallied independently but using the same encryption key pair. This is already possible with the current implementation of Helios. Another possible scenario is to run several different elections in *parallel* as long as no tallying of any election takes place before the last vote has been cast in all elections.

If one wishes to reuse the same key for k elections in *sequence*, the security in our proof will essentially need to ask k parallel decryption queries in sequence (each query corresponding to decrypting the dishonestly created ballots for one election). In such a scenario it is also important that one keeps copies of previous bulletin boards available and rejects any ballot that ever appeared on a previous board: an adversary could otherwise completely break the privacy of a honest user in one election if he can set up a second election in which the only ballot submitted is the ballot from the first election that he would like to decrypt.

With this duplicate-checking rule in place, the proof of security for minivoting that we gave in Section 4 can be adapted for the setting of k elections run with the same trustee keys, provided that the encryption is IND-k-CPA. In particular, if the encryption scheme is IND-CCA2 like the one proposed for Helios at Esorics [9], one could reuse the same key for polynomially many elections. Conversely, a voting scheme with ballot privacy for the result function that reveals all votes even when reusing keys up to k times can be shown to give an IND-k-CPA secure encryption scheme. In the security reductions, one simply substitutes "parallel decryption queries" for "tallying" and vice versa.

In the previous section we have shown that the Enc+FSPoK construction is IND-1-CPA. A natural question is whether the proof extends to IND-k-CPA. The rewinding technique that we used does not apply in this more general case, but the rewinding suggested by Shoup and Genaro [21] should allow to extend the proof to $k \in O(\log \lambda)$. This should come as no surprise as the separate parallel queries lead to nested rewindings (hence the logarithmic bound). The question is however wide open for k superlogarithmic in the security parameter.

References

1. Ben Adida. Helios: Web-based Open-Audit Voting. In *17th USENIX Security Symposium*, pages 335–348, 2008. Helios website: `http://heliosvoting.org`.

2. Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In T. Moran D. Jefferson, J.L. Hall, editor, *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. Usenix, August 2009.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - Proceedings of CRYPTO'98*, pages 26–45, Santa-Barbara, CA, USA, 1998. Springer-Verlag - LNCS Vol. 1462.
4. Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2003.
5. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 390–399, 2006.
6. Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer, 1999.
7. Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, January 1987.
8. Josh Benaloh. Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2006. USENIX Association.
9. David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot secrecy. In Springer, editor, *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11)*, volume 6879 of *LNCS*, 2011.
10. Philippe Bulens, Damien Giry, and Olivier Pereira. Running mixnet-based elections with Helios. In H. Shacham and V. Teague, editors, *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. Usenix, 2011.
11. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.
12. Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded cca2-secure encryption. In *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2007.
13. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.
14. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 542–552. ACM, 1991.
15. Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2001.
16. Jens Groth. Evaluating security of voting schemes in the universal composability framework. In *Applied Cryptography and Network Security—ACNS '04*, volume 3089 of *Lecture Notes in Computer Science*, pages 1–18, 2004.
17. Laurie Haustenne, Quentin De Neyer, and Olivier Pereira. Elliptic Curve Cryptography in JavaScript. In G. Leander and F.-X. Standaert, editors, *ECRYPT Workshop on Lightweight Cryptography*, November 2011.
18. R. Küsters, T. Truderung, and A. Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *IEEE Symposium on Security and Privacy (S&P 2011)*, pages 538–553. IEEE Computer Society, 2011.
19. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM, 1990.
20. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.
21. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.
22. Douglas Wikström. Simplified submission of inputs to protocols. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *Security and Cryptography for Networks, 6th International Conference*, volume 5229 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2008.

A Security notions for asymmetric encryption

In this section we recall some of the security notions for encryption that are relevant for this paper. Our presentation of existing notions follows the paper of Bellare, Desai, Pointcheval and

Rogaway [3] from Crypto '98 and the paper of Bellare and Sahai [6] from Crypto '99 in the latest revision of '06. This revision contains a number of improvements; for some of these the authors give credit to Lindell.

Indistinguishability is based on a thought experiment in which an adversary can pick any two messages of his choice, get an encryption of one of them and should not be able to tell which one was encrypted. This implies many intuitive properties including that an adversary seeing only a ciphertext should not be able to extract the plaintext. Indistinguishability is formulated by a heirarchy of three games, known as IND-CPA, IND-CCA1 and IND-CCA2 respectively and which differ in when the adversary can additionally obtain decryptions of ciphertexts: never, before he gets his "challenge ciphertext" or at any time, provided he does not ask for the challenge itself to be decrypted.

Non-malleability, originally formulated by Dolev, Dwork and Naor [14], asks that an adversary who sees a ciphertext of one of several messages of his choice cannot come up with further ciphertexts whose messages are "meaningfully" related to his challenge ciphertext, even if he cannot decrypt any of these himself. The precise formulation of non-malleability is quite involved; we refer the reader to the papers mentioned above. Again there are several possible definitions based on the details of the "meaningful relation" that the adversary should establish and on when he can obtain decryptions of other ciphertexts. There is again a heirarchy of NM-CPA, NM-CCA1 and NM-CCA2.

Bellare and Sahai [6] established that non-malleability is in fact equivalent to a form of indistinguishability in which the adversary gets an extra so-called "parallel decryption query". This may contain any number of ciphertexts all of which will be decrypted for the adversary (except if he ask for the challenge ciphertext itself to be decrypted), but he can only ask this query once. The term *parallel* reflects the fact, as Bellare and Sahai explain, that the adversary cannot choose one of the ciphertexts in this query after he has seen the decryption of a previous one. These notions are called IND-PCA0, IND-PCA1 and IND-PCA2 (the authors chose PCA0 over PCPA). In fact Bellare and Sahai showed that indistinguishability with one parallel decryption query is equivalent to two different notions of non-malleability, so-called comaprison- and simulation-based notions.

We propose further notions in which the adversary may ask up to a fixed number k of parallel decryption queries. As we explain in the paper IND-k-CPA security enables the reuse of authority keys across k elections. We define notions IND-k-χ where $k \in \mathbb{N}$ is the number of parallel decryption queries an adversary may make and $\chi \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$ describes the "regular" decryption queries available to the adversary. For $k = 0$ these notions are exactly the classical IND ones and for $k = 1$, the IND-P ones.

As an example we mention that Bellare and Sahai showed IND-PCA0 = NM-CPA (our IND-1-CPA) but a counter-example in their previous paper to separate IND-CPA and IND-CCA1 also shows that IND-1-CPA and hence NM-CPA is strictly weaker than IND-2-CPA. In other words, non-malleability buys you exactly one parallel decryption query or "degree of adaptivity" but not two. We formalize simultaneously these notions using the game defined below.

Setup. The challenger picks a bit $\beta \xleftarrow{R} \{0,1\}$ and runs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. The adversary is given pk but not sk.

Query phase 1. The adversary may query oracles $\mathcal{O}^{\mathsf{Dec}}$ and $\mathcal{O}^{\mathsf{P\text{-}Dec}}$ with ciphertexts of his choice. Their behaviour depends on the precise security property under consideration.

Challenge. The adversary may submit a pair of messages (m_0, m_1) to the challenger who responds by creating a challenge ciphertext $c^* \leftarrow \mathsf{Enc}(m_\beta)$ and returning this to the adversary.

Query phase 2. Like query phase 1 but the oracles may operate differently. In addition, the adversary is not allowed to use c^* in his inputs to either oracle (if he does, they simply return $\perp$).

Guess. At any time, the adversary may end the game by returning a bit α.

The oracles $\mathcal{O}^{\mathsf{Dec}}$ and $\mathcal{O}^{\mathsf{P\text{-}Dec}}$ operate as follows when active.

- $\mathcal{O}^{\mathsf{Dec}}$ takes a ciphertext c and returns $m \leftarrow \mathsf{Dec}(sk, c)$.
- $\mathcal{O}^{\mathsf{P\text{-}Dec}}$ takes a vector of ciphertexts $\mathbf{c}$, decrypts each one individually using sk and returns the vector $\mathbf{m}$ of messages.

When inactive, they return $\perp$ for any input. Whether or not they are active depends on the precise security property.

The security properties modeled are denoted by

$$\text{IND-}k\text{-}\chi \qquad k \in \mathbb{N},\ \chi \in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$$

A value of $k = 0$ can be dropped, thus IND-0-CPA is just written IND-CPA.

The oracle $\mathcal{O}^{\mathsf{Dec}}$ is active as follows:

phase / χ	CPA	CCA1	CCA2
query phase 1	inactive	active	active
query phase 2	inactive	inactive	active

The oracle $\mathcal{O}^{\mathsf{P\text{-}Dec}}$ is active only in the second query phase and only for k queries, after which it deactivates itself. For example, IND-2-CCA1 denotes the game in which $\mathcal{O}^{\mathsf{Dec}}$ is active in phase 1 but not phase 2 and $\mathcal{O}^{\mathsf{P\text{-}Dec}}$ is active in the second query phase for at most 2 calls.

Definition A.1 (IND Security). *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is secure with respsect to notion* IND-k-χ *if for any efficient adversary A, his advantage $\mathbf{Adv}_{A,C}(\lambda)$ is negligible as a function of λ where C is the challenger in the game described above and the oracles are as described for the given notion.*

Relations among security notions. Bellare, Desai, Pointcheval and Rogaway [3] studied the relations between the IND(-0) notions and NM notions. Bellare and Sahai [6] introduced and studied the notions where, in our notation, $k = 1$.

$$\begin{tikzcd}
\text{IND-}k\text{-CPA} \arrow[d, "\cdots"] & \text{IND-}k\text{-CCA1} \arrow[l] \arrow[d, "\cdots"] & \text{CCA2} \arrow[l, "\mathit{all}"'] \\
\text{IND-1-CPA} & \text{IND-1-CCA1} \arrow[l] & \\
= \text{IND-PCA0} & = \text{IND-PCA1} & \\
= \text{NM-CPA} \arrow[d] & = \text{NM-CCA1} \arrow[d] & \\
\text{IND-0-CPA} & \text{IND-0-CCA1} \arrow[l] & \\
= \text{IND-CPA} & = \text{IND-CCA1} &
\end{tikzcd}$$

Theorem A.2. *The above diagram captures exactly the relations between security notions.*

1. *For any pair of notions A, B with an arrow $A \longrightarrow B$, any public-key encryption scheme secure against notion A is also secure against notion B.*
2. *For any pair of notions C, D with no path from C to D, if there is a scheme secure with respect to notion D then there is also a scheme that is secure with respect to D but not with respect to C.*
3. *For any pair of notions with an equals sign between them (and also for any two notions ending in CCA2) a scheme is secure with respect to one notion if and only if it is secure with respect to the other.*

The IND-0 = IND equivalences are obvious as both notions denote the same game. Similarly the IND-P notions are identical to our IND-1 notions. All IND-k-CCA2 notions are equivalent because one can always simulate "missing" parallel decryption queries using the regular decryption oracle which is available at all times and an unlimited number of times in the IND-CCA2 game.

The arrow-directions are obvious too: for any attack scenario χ and any $k > 0$, a valid adversary making at most $k - 1$ queries is also valid for the game that allows him k queries. Similarly, a CPA adversary is also a valid CCA1 adversary and a CCA1 adversary is also a valid CCA2 adversary.

The IND-P = IND-1 equivalences were proven by Bellare and Sahai [6]. The cases $k \leq 1$ were studied by Bellare, Desai, Pointcheval and Rogaway [3]; they proved both the implications and the separations. In particular they proved IND-CCA2 = NM-CCA2.

The only remaining cases are the separations involving IND-k-{CPA,CCA1}. For these, we turn to the counter-examples provided in the aforementioned paper [3].

To show that NM-CPA does not imply IND-CCA1, Bellare et al. first try to start with an NM-CPA secure scheme, pick a bitstring not in the image of Enc and define the decryption of this string to be the secret key. This string is then provided to the adversary as part of the public key. The idea is that an adversary with no decryption queries can do no better given this string than without it but a single decryption query will totally break the scheme.

Bellare et al. did note however that they could neither prove or disprove that this scheme remains NM-CPA secure. Using the later paper of Bellare and Sahai [6], we can show that it is not: as the scheme falls to a single decryption query, it is not IND-1-CPA, hence not NM-CPA either.

The correct counterexample provided by Bellare et al. uses an extra indirection. The adversary is provided with a string u in the public key that "decrypts" to some other string v, both not in the image of Enc. Decrypting v now yields the secret key. Note that the adversary must make two decryption queries in sequence and requires the result of the first before launching the second. Bellare et al. showed that this scheme remains NM-CPA but it is evidently not IND-CCA1.

We note that it is IND-1-CPA (and so NM-CPA) but not IND-2-CPA. This also gives us our counterexample for any k of a scheme that is IND-k-CPA but not IND-$k+1$-CPA.

Lemma A.3. *For any $k \in \mathbb{N}$, if there is an IND-k-CPA secure encryption scheme then there is also one that is IND-k-CPA but not IND-$(k+1)$-CPA.*

Proof. Let an IND-k-CPA secure scheme be given. We pick a set S of size at least 2^λ whose intersection with the image of Enc is empty and a chain of $k+1$ uniformly and randomly chosen strings s_i from S. We define a new scheme by taking the given one and letting $\mathsf{Dec}(sk, s_i) = s_{i+1}$ for $i \leq k$ and $\mathsf{Dec}(sk, s_{k+1}) = sk$. We provide s_1 in the public key.

For any adversary against our scheme, we reduce to IND-k-CPA security of the given one by picking $k+1$ random elements $(s_i)_{i=1}^{k+1}$ from S and providing s_1 together with the public key. Whenever the adversary asks a decryption query using an s_i for $i \leq k$, we return s_{i+1} as the decryption. If the adversary asks to decrypt s_{k+1} (before he has exhausted all his queries) we abort the reduction.

As long as we do not abort, the view of the adversary in the reduction is identically distributed to that against our new encryption scheme and his advantage translates directly into ours against IND-k-CPA for the original scheme.

If we have to abort, consider the first time the adversary asked a decryption query containing an s_i for $i > 1$ for which he had not already asked s_{i-1} to be decrypted. It may or not be the case that this element is s_{k+1} itself but as the adversary has at most k decryption queries, such an event must have occurred.

This means the adversary must have guessed s_i as it was uniformly random in S (minus the other k elements) but $|S| - k \geq 2^\lambda - k$. The probability of such a guess being correct is at most $1/\left(2^\lambda - k\right)$ which is negligible in λ (note that k is a constant). It follows that the probability of an adversary forcing our reduction to abort with at most k queries is negligible. Therefore, the scheme is still IND-k-CPA secure. The scheme is clearly not IND-$(k+1)$-CPA secure however as a sequence of $k+1$ adaptive decryption queries on the extra strings yields the secret key. $\square$