A Blind Coupon Mechanism Enabling Veto Voting over Unreliable Networks

Colin Boyd Kristian Gjøsteen Clémentine Gritti
Thomas Haines

NTNU, Trondheim, Norway
{colin.boyd,kristian.gjosteen,clementine.gritti,thomas.haines}@ntnu.no

Abstract

A Blind Coupon Mechanism (BCM) allows spreading of alerts quietly and quickly over unreliable networks. The BCM primitive ensures that alerts are efficiently broadcast while the nature of these signals are securely hidden. However, current BCM proposals are limited to indicating a single bit of information and also lack natural privacy properties. In this paper, we develop a new BCM solution that allows the transmission of several alerts privately and in one shot. This extension perfectly suits a new kind of applications, that is (absolute) veto voting over unreliable networks, in which multiple decisions are reached using only simple peer-to-peer communications. Our enhanced BCM efficiently supports the spread of votes over unreliable networks while hiding whether these votes contain any or several vetoes. We prove our BCM solution secure and illustrate its use for veto voting protocols in limited communication infrastructures.

Keywords: Blind coupon mechanism, Veto voting system, Untraceability.

1 Introduction

A *Blind Coupon Mechanism* (BCM) [2, 3, 5] is a cryptographic primitive allowing confidential signals to be combined in a specific way. A BCM enables a network to distribute a covert signal while ensuring that forging signals is difficult. Signals are embedded in coupons, which are generated by authorities in possession of some secret material. Dummy coupons (without a signal) and signal coupons are exchanged over the network, and combined such that anything joined with a signal coupon becomes a signal coupon.

Such a primitive can prevent an attacker from gaining knowledge about alert transmissions. Suppose an attacker manages to access a network and monitor a sensor-based intrusion detection system. The attacker knows that

she is safe until the sensors transmit an alert. Therefore, the system requires a confidential and efficient alert transmission to prevent the attacker from withdrawing and removing her tracks in a timely manner.

The BCM notion was first introduced by Aspnes et al. [2, 3] and recently improved by Blazy and Chevalier [5]. A BCM includes a *verification algorithm* that checks whether a coupon is valid, and a *combining algorithm* that takes as inputs two valid coupons and outputs a coupon. The output coupon is signal (with high probability) if and only if at least one of the inputs is signal. A BCM should satisfy two security properties: *indistinguishability* (an attacker cannot distinguish between dummy and signal coupons) and *unforgeability* (an attacker cannot create a signal coupon unless it has another signal coupon as input).

1.1 Generalized BCMs

Previously proposed BCMs use coupons which can only transmit one bit of information, for example whether an alert has occurred or not. When we consider using BCMs in different applications we may want to have coupons which can signal multiple events. Even in the original application of quietly spreading alerts, it may be useful to be able to indicate additional information on which kind of alert is relevant. We therefore propose a generalized definition of BCMs allowing multiple signals. A trivial way to instantiate our generalized BCM is to use multiple instances of a standard BCM, but we would like also to remain as efficient as possible. Therefore we proposed a construction for a generalized BCM which adds minimally to the overhead for the currently most efficiently known BCM construction.

We are also interested in stronger security for BCMs. Previous BCM analysis [2, 3, 5] considers only indistinguishability and unforgeability as relevant security properties, ignoring the privacy of the agents contributing coupons. Malicious spotters may attempt to track and follow coupons and thus gain information on individual choices. The original BCM construction [2, 3] even suffers from the creation of traceable coupons if key generation is dishonest. Hence, there is a clear lack of privacy notion in [2, 3, 5], and we therefore add the notion of untraceability to the useful security properties of BCMs.

As a useful new application for our generalized BCM, we propose a novel voting protocol. One way to look at a BCM is that signal coupons override dummy coupons – once a signal coupon has been added, the dummy coupons are irrelevant. In a sense this means that the signal coupons *veto* the effect of dummy coupons. This observation suggests that a BCM is a natural foundation for (absolute) veto voting [17, 14, 4]. In general, voting systems are designed for elections launched in reliable communication infrastructures to support interaction with central servers. Nevertheless, in some situations, only limited connectivity is available and continuous and

local tallying is needed. These restrictive communication infrastructures require to broadcast vetoes quietly and quickly, while the nature of these votes must be securely hidden, allowing us to exploit the untraceability property of our generalized BCM. We depict below a plausible scenario that requires a voting system that fully operates under such conditions.

One could assume that communication channels between system users are secure. Nevertheless, we must defend the system against dishonest participants, who may not forward the received coupons or being curious about others' choices. Hence such an assumption will not help. One could also attempt to encrypt veto coupons and return them to the authorities via peer-to-peer channels, but that would require solving problems related to traceability and privacy similar to those needed in designing a BCM.

1.2 A BCM-Friendly Voting Scenario

A political demonstration is calling for a change of government. The organizers wish demonstrators to express their choices on elements of their new plan. Elements could be whether a demonstration should be scheduled on the coming Saturday. Several parameters may disrupt the voting process and influence the underlying system. For instance, the government may respond to the demonstration by either turning off the mobile phone network or jamming the usable radio frequencies, making radio communication impossible. Although QR-codes with cameras enable a low-bandwidth peer-to-peer communication, the latency is too large to allow reasonable ad-hoc networks. Therefore, the system requires other tools to overcome such communication constraints.

Furthermore, the government must not be able to infiltrate the demonstration network, and to forge a veto canceling any further demonstration. In addition, the organizers running the veto elections may move continuously, and therefore, should sample current veto states from any physical location in the demonstration at any time. Not all demonstrators trust the organizers, and information on who vetoed what may have bad consequences (e.g. being registered in police files as a potential governmental opponent). Hence, privacy and untraceability properties must be guaranteed.

The above scenario motivates the design of a veto voting system in unreliable network infrastructures. During a setup phase, the organizers, seen as *election authorities*, distribute ballots to demonstrators, seen as *voters*. These ballots are pre-marked as either blank or carrying veto(es). During a voting phase, the demonstrators interact using limited peer-to-peer communications, and make their choices, for instance regarding demonstration cessation. They are assured to remain anonymous, hence avoiding political friction among them. Once the voting phase is over, the organizers sample votes by interacting with the demonstrators, and recover the veto results.

Such a veto voting system will enable demonstrators to veto even under

the following attacks or constraints from the government. The system will continue to operate successfully, even if the government disturbs the network. Attackers may also attempt to infiltrate the network; however, as an outsider, it will not be able to disrupt the voting process.

1.3 Proposed Solution

Our BCM solution is based on that of Blazy-Chevalier [5] which offers attractive features of strong indistinguishability, unforgeability and applicability to real contexts. We enhance this BCM to allow the transmission of multiple signals in one coupon, only adding one extra group element per signal. We also define a new security property, namely *untraceability*, and prove that our solution satisfies it. This property precludes an attacker to track, follow and distinguish transmitted coupons.

In addition, we show that veto elections arise as a natural application of BCM. We design a peer-to-peer veto voting system based on our improved BCM, enabling the spread of coupons containing (possibly multiple) veto(es).

- During the setup phase, dummy coupons, representing a *blank* vote, and signal coupons, representing a *veto* vote, are distributed to voters by the authorities via secure communication channels (this is the only phase where such feature must be guaranteed).

- The voters first spread their dummy coupons. Communication channels no longer need to be secure. This is to create a continuous flow of coupons during the entire voting phase, and prevent attackers to distinguish blank and veto votes from a possible discontinuity in spreading them. Whenever a voter decides to veto on some action(s), she uses the appropriate signal coupon and spreads it around her.

- Voters continuously exchange their coupons with their neighbors, ensuring the aforementioned continuous flow. Upon receiving a neighbor's coupon, a voter combines it with her current coupon, and gets an updated coupon, that is then spread around her. One fundamental property of our veto election is that an output coupon is signal (veto) if at least one input is signal. Moreover, one veto is enough to stop an action, and thus vote counting is not required.

- The election authorities intercept coupons at random time from random voters and decode them to obtain veto results. Once a veto is received, this is enough to unilaterally stop the associated action.

Figure 1 illustrates our veto voting system based on BCM. Several election authorities jointly create and distribute initial coupons to the voters. Voters form a network where peer-to-peer communications are possible but

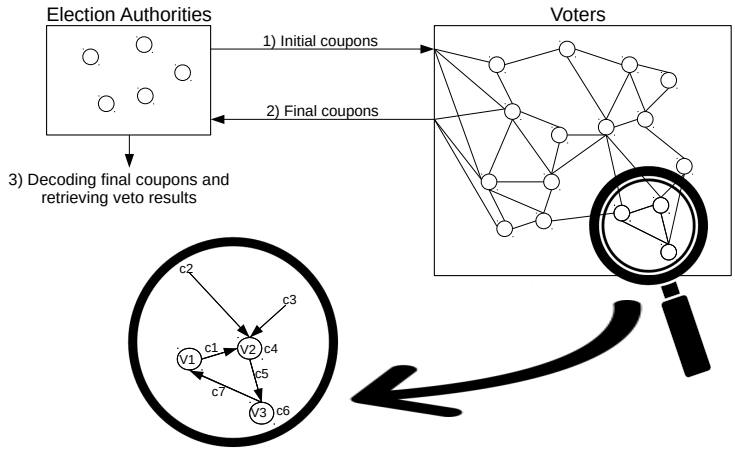

Figure 1: Veto voting system based on BCM.

limited. Voters spread their coupons around them such that the resulting flow is continuous. From the magnifying glass, we observe that the voter $V2$ has collected coupons $c1$, $c2$ and $c3$ from other voters. She wishes to use her coupon $c4$ for voting. She then emits coupon $c5$ to her neighbors, that is a combination of $c1$, $c2$, $c3$ and $c4$. The voter $V3$ has received coupon $c5$ and combines it with his own coupon $c6$, and spreads the resulting coupon $c7$ around him. Once the voting phase is over, the election authorities intercept final coupons from the voters and jointly decode them to obtain the veto results.

1.4 Related Work

Blind Coupon Mechanism. Aspnes et al. [2, 3] introduce BCM as an AND-homomorphic authenticated bit commitment scheme. The authors construct their scheme based on an abstract group structure (U, G, D) such that U is a finite set, $G \subseteq U$ is a cyclic group and D is a proper subgroup of G. The elements in D are dummy coupons and the elements in $G \setminus D$ are signal coupons. The scheme is proved secure with relation to indistinguishability and unforgeability based on the subgroup membership and subgroup escape problems respectively. The subgroup escape problem [2, 3] is defined as follows: Given a generator for D (but not G), find an element of $G \setminus D$. The problem seems hard on certain groups with bilinear pairings and on elliptic curves over the ring $\mathbb{Z}_n$.

Recently, Blazy and Chevalier [5] propose a more efficient and more secure scheme compared to that of Aspnes et al. [2, 3] by setting their scheme in a group of prime order instead of composite order, and by relying on standard assumptions. They design a simple BCM scheme that is OR-homomorphic and combine this scheme with a new version of the linearly homomorphic Structure-Preserving Signature (SPS) scheme [18] to

obtain their full, AND-homomorphic BCM scheme. Blazy-Chevalier scheme is proved indistinguishable under the Decisional Diffie-Hellman assumption and is statistically unforgeable.

Nevertheless, implementing a veto protocol over constrained networks using Blazy-Chevalier BCM technique generates computational and communication burdens and impedes its operation. Since we aim to develop veto voting in restrictive peer-to-peer communication infrastructures, we need to extend the aforementioned solution to permit secure and efficient multiple veto broadcast and guarantee practical performance.

Veto Voting. In a reformulation of the famous *Dining Cryptographers* (DC) problem introduced by Chaum [9], Hao and Zieliński [14] consider the following problem. Three cryptographers wish to know among NSA and them, who has not paid for the dinner, such that all the participants are enabled to pay anonymously. If one participant votes with a veto, then one of the cryptographers has paid for the dinner; otherwise, NSA has paid. A protocol for the statement that no cryptographer has paid is thus similar to anonymous veto protocols [17, 13, 7]. Hao and Zieliński [14] present a protocol, named *Anonymous Veto Network* (AV-net), to solve the reformulated problem. Compared to the DC network solution [9], AV-net does not require secret channels, does not encounter message collisions and is more resistant to disruptions.

The Kiayias-Yung protocol [17] for veto elections considers a tally without any veto votes as a 0-vote. A voter who wants to veto actually votes on a non-zero random element from $\mathbb{Z}_p$, for a prime p, and if no voter has vetoed, then the tally is 0. One security issue is that any voter can check whether she is the only one who vetoed using her random element [13]. Groth [13] improves the Kiayias-Yung protocol while publishing a smaller amount of data. However, the number of rounds depends on the number of voters, while it is constant in the original Kiayias-Yung protocol [17]. Brandt [7] bases his veto protocol on a technique used for secure multi-party computation applications. While it allows to define a boolean-OR function, solving the function remains complex and expensive.

Hao and Zieliński's AV-net [14] outperforms existing anonymous veto protocols [17, 13, 7] regarding the number of rounds, computational load and bandwidth usage. Nevertheless, the solution [14] still requires two rounds on voters' side: a first round to send a seed with its proof (preparation phase), and a second round to actually vote (voting and self-tallying phase). To reduce to one round, the preparation phase should be executed only once [11]: voters agree on some parameters before several elections, and use these parameters for all the elections they are participating in.

AV-net [14] also suffers from two issues related to fairness and robustness. First, a voter who has submitted a veto can find out whether there are any

other voters who vetoed. Second, the last voter submitting a vote is able to pre-compute the boolean-OR result before submission, allowing the last voter to change the vote according to that pre-computation. Khader et al. [16] propose a variant Hao-Zieliński protocol [14] to provide aforementioned missing properties (by adding a commitment round and a recovery round respectively). Their variant also assumes authenticated public channels to prevent multiple voting and to guarantee voters eligibility. Later, Bag et al. [4] extend further to avoid the fairness and robustness limitations of the previous work, but now maintaining the advantage of a two-round protocol. Indeed, Bag et al.'s solution [4] achieves similar system complexities to AV-net [14], while binding voters to their votes in the very first round, canceling the possibility of runtime changes to any of the inputs. In addition, at the end of the voting phase, voters are not able to learn more than the output of the boolean-OR function and their own votes.

We see that earlier veto voting systems need fairly reliable networks or a sequential round structure in order to work. Neither is available in our scenarios, where connectivity is unreliable and we need continuous and local tallying.

1.5 Contributions

Our paper contains contributions to the design of both a secure BCM scheme and a veto voting system. More precisely, we provide a new BCM construction (Section 3) that allows for transmitting multiple signals in one coupon, rather than in separate coupons. We prove it secure according to indistinguishability and unforgeability properties. We also define the notion of untraceability, as an enhancement of the privacy in BCM, and prove that our solution satisfies it. Then, we show that veto voting arises as a straight application of BCM (Section 4). We present a peer-to-peer veto voting protocol over unreliable networks based on our solution.

2 Building Blocks

2.1 Preliminaries

In this section, we introduce the mathematical tools and assumptions that our BCM construction and security proofs use.

Bilinear Group and Pairing. Our extended BCM relies on pairing-based cryptography. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. A pairing e is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which satisfies the following properties:

- Bilinearity: Given $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;

- Non-degeneracy: There exist $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$;
- Computability: There exists an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

n-Decisional Diffie-Hellman Exponent (n-DDHE) Assumption. Our scheme is proven untraceable assuming that the n-DDHE problem is hard. Let $\mathbb{G}_1$ be a cyclic group of prime order p and g_1 be a generator of that group. The DDHE assumption [6, 15] compares the real and random distributions:

$$\mathsf{DHE_n} = \{\{g_1^{\nu^i}\}_{1\leq i\leq n}, g_1^{\nu^{n+1}}; \nu \in_R \mathbb{Z}_p\} \quad \mathsf{DHE_n^\$} = \{\{g_1^{\nu^i}\}_{1\leq i\leq n}, g_1^{\mu}; \nu, \mu \in_R \mathbb{Z}_p\}$$

A (T, ε)-distinguisher for $\mathbb{G}_1$ is a probabilistic Turing Machine Δ running in time T that, given an element X of either $\mathsf{DHE_n}$ or $\mathsf{DHE_n^\$}$, outputs 0 or 1 such that:

$$Adv_{\mathbb{G}_1}^{ddhe_n}(\Delta) = |\Pr[\Delta(X) = 1, X \in \mathsf{DHE_n}] - \Pr[\Delta(X) = 1, X \in \mathsf{DHE_n^\$}]| \geq \varepsilon$$

The DDHE problem is (T, ε)-intractable if no (T, ε)-distinguisher for $\mathbb{G}_1$ exists.

n-Multi-Decisional Diffie-Hellman (n-MDDH) Assumption. Our scheme is proven indistinguishable assuming that the n-MDDH problem is hard. Let $\mathbb{G}_1$ be a cyclic group of prime order p and g_1 be a generator of that group. The MDDH assumption [8] compares the real and random distributions:

$$\mathsf{MDH_n} = \{g_1^{x_0}, \{g_1^{x_j}\}_{1\leq j\leq n}, \{g_1^{x_0 x_j}\}_{1\leq j\leq n}; x_0, x_j \in_R \mathbb{Z}_p, 1 \leq j \leq n\}$$
$$\mathsf{MDH_n^\$} = \{g_1^{x_0}, \{g_1^{x_j}\}_{1\leq j\leq n}, \{g_1^{r_{0,j}}\}_{1\leq j\leq n}; x_0, x_j, r_{0,j} \in_R \mathbb{Z}_p, 1 \leq j \leq n\}$$

A (T, ε)-distinguisher for $\mathbb{G}_1$ is a probabilistic Turing Machine Δ running in time T that, given an element X of either $\mathsf{MDH_n}$ or $\mathsf{MDH_n^\$}$, outputs 0 or 1 such that:

$$Adv_{\mathbb{G}_1}^{mddh_n}(\Delta) = |\Pr[\Delta(X) = 1, X \in \mathsf{MDH_n}] - \Pr[\Delta(X) = 1, X \in \mathsf{MDH_n^\$}]| \geq \varepsilon$$

The MDDH problem is (T, ε)-intractable if no (T, ε)-distinguisher for $\mathbb{G}_1$ exists.

Double Pairing (DP) Assumption. Our scheme is proven unforgeable assuming that the DP problem is hard. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. The DP assumption [1] is defined as follows. A (T, ε)-adversary for $\mathbb{G}_1$ and $\mathbb{G}_2$ is a probabilistic Turing Machine Δ running in time T that given

a random pair $X = (g_2, \hat{g}_2)$ in $\mathbb{G}_2^2$, outputs a non-trivial pair $(g_1, \hat{g}_1)$ in $\mathbb{G}_1^2$ satisfying $Y : e(g_1, g_2) \cdot e(\hat{g}_1, \hat{g}_2) = 1_{\mathbb{G}_T}$ such that:

$$Adv^{dp}_{\mathbb{G}_1,\mathbb{G}_2}(\Delta) = |\Pr[X = (g_2, \hat{g}_2) \in \mathbb{G}_2^2; (g_1, \hat{g}_1) \leftarrow \Delta(X) : (g_1, \hat{g}_1) \in \mathbb{G}_1^2 \wedge Y]| \geq \varepsilon$$

The DP problem is (T, ε)-intractable if no (T, ε)-adversary for $\mathbb{G}_1$ and $\mathbb{G}_2$ exists.

2.2 Linearly Homomorphic Structure-Preserving Signature

A linearly homomorphic SPS scheme [18] enables verifiable computation mechanisms on encrypted data, by combining homomorphic signature properties with the additional one that signatures and messages only contain group elements with unknown discrete logarithms. Blazy and Chevalier [5] present a linearly homomorphic SPS scheme in an asymmetric setting with groups of prime order, as an extension of the scheme in the symmetric setting given in [18].

Following the idea of Blazy and Chevalier [5], a one-time linearly homomorphic SPS scheme is combined with an OR-homomorphic BCM in the asymmetric setting to guarantee unforgeable signal coupons (the one-time property implies that the tag linked to the to-be-signed message is empty). We extend this combination to the multivariate setting by signing vectors of $n + 1$ elements rather than signing vectors of two elements.

For clarity, we recall the signature scheme introduced in [18]. The one-time linearly homomorphic SPS scheme is composed of the following algorithms:

$\mathsf{KeyGen}(1^k) \rightarrow (pk, sik)$**.** On input of a security parameter 1^k, the algorithm outputs a verification (public) key pk and a signing (secret) key sik.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a tuple defined from the bilinear group setting. Pick at random $\alpha \in_R \mathbb{Z}_p$ and compute $\hat{g}_2 = g_2^{\alpha}$. For $j \in [0, n]$, pick at random $\xi_j, \rho_j \in_R \mathbb{Z}_p$ and compute $hp_j = g_2^{\xi_j} \hat{g}_2^{\rho_j}$. The verification key is $pk = (g_1, g_2, \hat{g}_2, \{hp_j\}_{j\in[0,n]})$ and the signing key is $sik = (\{\xi_j, \rho_j\}_{j\in[0,n]})$.

$\mathsf{Sign}(sik, \vec{m}) \rightarrow \sigma$**.** On input of a signing key sik and a message vector $\vec{m} = (m_0, m_1, \cdots, m_n) \in \mathbb{G}_1^{n+1}$, the algorithm outputs a signature σ. Compute $z = \prod_{j=0}^{n} m_j^{-\xi_j}$ and $u = \prod_{j=0}^{n} m_j^{-\rho_j}$ and set the signature $\sigma = (z, u)$.

$\mathsf{SignDerive}(pk, \{\lambda_i, \sigma_i\}_{i\in[1,l]}) \rightarrow \sigma$**.** On input of a verification key pk and l pairs of coefficients (arbitrary) and signatures $\{\lambda_i, \sigma_i\}_{i\in[1,l]}$,the algorithm outputs a new signature σ. If all the input signatures are valid, compute $z = \prod_{i=1}^{l} z_i^{\lambda_i}$ and $u = \prod_{i=1}^{l} u_i^{\lambda_i}$ and set the new signature $\sigma = (z, u)$.

$\mathsf{Verify}(pk, \vec{m}, \sigma) \rightarrow \{\mathtt{valid}, \mathtt{invalid}\}$**.** On input of a verification key pk, a message vector $\vec{m} = (m_0, m_1, \cdots, m_n) \in \mathbb{G}_1^{n+1}$ and a signature $\sigma = (z, u)$, the algorithm outputs `valid` or `invalid`. The signature is valid if and only if $e(z, g_2) \cdot e(u, \hat{g}_2) \cdot \prod_{j=0}^{n} e(m_j, hp_j) = 1_{\mathbb{G}_T}$ holds.

Correctness. For all key pair $(pk, sik) \leftarrow \mathsf{KeyGen}(1^k)$, for all message vector $\vec{m}$, if $\sigma \leftarrow \mathsf{Sign}(sik, \vec{m})$, then $\mathsf{Verify}(pk, \vec{m}, \sigma) = \mathtt{valid}$.

Let $\{\lambda_i, \vec{m}_i\}_{i\in[1,l]}$ correspond to $\prod_{i=1}^{l} \vec{m}_i^{\lambda_i} = (\prod_{i=1}^{l} m_{i,0}^{\lambda_i}, \prod_{i=1}^{l} m_{i,1}^{\lambda_i}, \cdots, \prod_{i=1}^{l} m_{i,n}^{\lambda_i})$. For all key pair $(pk, sik) \leftarrow \mathsf{KeyGen}(1^k)$, for all signatures σ_i on message vectors $\vec{m}_i$, if $\mathtt{valid} \leftarrow \mathsf{Verify}(pk, \vec{m}_i, \sigma_i)$ for $i \in [1, l]$, then $\mathtt{valid} \leftarrow \mathsf{Verify}(pk, \{\lambda_i, \vec{m}_i\}_{i\in[1,l]}, \mathsf{SignDerive}(pk, \{\lambda_i, \sigma_i\}_{i\in[1,l]}))$.

3 New Blind Coupon Mechanism

Our BCM solution extends Blazy-Chevalier scheme [5] into a multivariate setting as follows: in addition to the two elements $g_1, h_1 \in \mathbb{G}_1$, $n-1$ elements $h_2, \cdots, h_n$ are also generated, where n is an integer. This implies that a signal coupon is a tuple of $n+1$ random elements from the group $\mathbb{G}_1$, while a dummy coupon is a tuple $(g_1^r, h_1^r, h_2^r, \cdots, h_n^r)$ of $n+1$ elements where $r \in_R \mathbb{Z}_p^*$ is a random exponent.

3.1 Construction

The extended BCM construction contains the following algorithms:

$\mathsf{BCMGen}(1^k) \rightarrow (pk, sk)$**.** On input a security parameter 1^k, the algorithm outputs the public key pk and the secret key sk.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a tuple defined from the bilinear group setting. Run the algorithm $\mathsf{KeyGen}(1^k) \rightarrow (pk, sik)$ where $pk = (g_1, g_2, \hat{g}_2, \{hp_j\}_{j\in[0,n]})$ and $sik = (\{\xi_j, \rho_j\}_{j\in[0,n]})$. For $j \in [1, n]$, pick at random $\beta_j \in_R \mathbb{Z}_p$ and compute $h_j = g_1^{\beta_j}$. The public key is pk and the secret key is $sk = (sik, \{\beta_j\}_{j\in[1,n]}, \{h_j\}_{j\in[1,n]})$.

$\mathsf{BCMCouponGen}(pk, sk, \vec{v}) \rightarrow c$**.** On inputs the public key pk, the secret key sk, a vector $\vec{v} = (v_1, \cdots, v_n) \in \{0, 1\}^n$, the algorithm outputs a coupon c.

The integer n corresponds to the number of signal options. For $j \in [1, n]$, each element v_j of the vector $\vec{v}$ states the nature of the corresponding option opt_j. In particular, a bit $v_j = 0$ states that the option opt_j is set as dummy while a bit $v_j = 1$ states that it is set as signal. Hence, a dummy coupon is defined as $\vec{v} = (0, 0, \cdots, 0)$ (all 0s) and a coupon with signal on one single option is set as $\vec{v} = (0, \cdots, 0, 1, 0, \cdots, 0)$ (one 1 and the remaining 0s).

We recall that the secret key sk includes the signing key $sik = (\{\xi_j, \rho_j\}_{j\in[0,n]})$. To generate a valid coupon for a dummy or single signal coupon, pick at random $\delta, \delta' \in_R \mathbb{Z}_p$, and then compute $c_{1,0} = g_1^{\delta}$ and for $j \in [1, n]$, $c_{1,j} = h_j^{\delta+v_j\delta'} = g_1^{\beta_j(\delta+v_j\delta')}$. Set $c_1 = \{c_{1,j}\}_{j\in[0,n]}$.

Run the algorithm $\mathsf{Sign}(sik, c_1) \rightarrow c_2 = (z, u)$ where:

$$z = \prod_{j=0}^{n} c_{1,j}^{-\xi_j} = g_1^{-\xi_0\cdot\delta-\sum_{j=1}^{n}\xi_j\beta_j(\delta+v_j\delta')} \quad u = \prod_{j=0}^{n} c_{1,j}^{-\rho_j} = g_1^{-\rho_0\cdot\delta-\sum_{j=1}^{n}\rho_j\beta_j(\delta+v_j\delta')}$$

Let the valid coupon be $c = (c_1, c_2)$ where $c_1 = \{c_{1,j}\}_{j\in[0,n]}$ and $c_2 = (z, u)$. We call the pair $c_2 = (z, u)$ the *signature* of the coupon.

$\mathsf{BCMVerify}(pk, c) \rightarrow \{\texttt{valid}, \texttt{invalid}\}$**.** On inputs the public key pk and a coupon c, the algorithm outputs `valid` if the coupon c is valid; `invalid` otherwise.

Let a coupon be $c = (c_1, c_2)$ where $c_1 = \{c_{1,j}\}_{j\in[0,n]}$ and $c_2 = (z, u)$. Run $result \leftarrow \mathsf{Verify}(pk, c_1, c_2)$ and output $result$ (which is `valid` or `invalid`).

$\mathsf{BCMCombine}(pk, \{c_i\}_{i\in[1,l]}) \rightarrow c$**.** On inputs the public key pk, a set $\{c_i\}_{i\in[1,l]}$ of l valid coupons, the algorithm outputs a new valid coupon c combining c_i for $i \in [1, l]$.

Let $c_i = (\{c_{i,1,j}\}_{j\in[0,n]}, (z_i, u_i))$ be a coupon for $i \in [1, l]$. Run the algorithm $\mathsf{SignDerive}(pk, \{\lambda_i, (z_i, u_i)\}_{i\in[1,l]}) \rightarrow c_2$ for random exponents λ_i. That algorithm first checks that all the input coupons are valid. If the answer is positive, then it computes $z = \prod_{i=1}^{l} z_i^{\lambda_i}$ and $u = \prod_{i=1}^{l} u_i^{\lambda_i}$ and sets $c_2 = (z, u)$. Moreover, compute $c_{1,j} = \prod_{i=1}^{l} c_{i,1,j}^{\lambda_i}$ for $j \in [0, n]$, and set $c_1 = \{c_{1,j}\}_{j\in[0,n]}$. Finally, set the new, combined coupon $c = (c_1, c_2)$.

$\mathsf{BCMDecode}(pk, sk, c) \rightarrow \vec{v}$**.** On inputs the public key pk, the secret key sk and a valid coupon c, the algorithm outputs a vector $\vec{v} = (v_1, \cdots, v_n)$. The vector $\vec{v}$ reveals the nature of the coupon such that each element v_j of the vector $\vec{v}$ tells whether the corresponding option opt_j is either dummy or signal.

Given a coupon $c = (c_1, c_2)$ where $c_1 = \{c_{1,j}\}_{j\in[0,n]}$, if $c_{1,j} = c_{1,0}^{\beta_j}$ then set $v_j = 0$ for each $j \in [1, n]$; otherwise set $v_j = 1$. Output $\vec{v} = (v_1, v_2, \ldots, v_n)$.

Correctness on Coupon Decoding. Correctness on coupon decoding focuses on correctness of signature verification. The remaining is easily proved correct. For all key pair $(pk, sk) \leftarrow \mathsf{BCMGen}(1^k)$, for all vector $\vec{v}$, if $c \leftarrow \mathsf{BCMCouponGen}(pk,\ sk, \vec{v})$, then $\texttt{valid} \leftarrow \mathsf{BCMVerify}(pk, c)$ and

$\vec{v} \leftarrow \mathsf{BCMDecode}\ (pk, sk, c)$. Let $c = (c_1, c_2) = (\{c_{1,j}\}_{j\in[0,n]}, (z, u))$ be a valid coupon, then we have:

$$e(z, g_2)e(u, \hat{g}_2)\prod_{j=0}^{n} e(c_{1,j}, hp_j) = e(\prod_{j=0}^{n} c_{1,j}^{-\xi_j}, g_2)e(\prod_{j=0}^{n} c_{1,j}^{-\rho_j}, \hat{g}_2)\prod_{j=0}^{n} e(c_{1,j}, g_2^{\xi_j}\hat{g}_2^{\rho_j}) = 1_{\mathbb{G}_T}.$$

Correctness on Combined Coupon Decoding. For all key pair $(pk, sk) \leftarrow \mathsf{BCMGen}(1^k)$, for all coupons c_i for $i \in [1, l]$, if $\mathtt{valid} \leftarrow \mathsf{BCMVerify}(pk, c_i)$, then

$$\mathsf{BCMDecode}(pk, sk, \mathsf{BCMCombine}(pk, \{c_i\}_{i\in[1,l]})) = \bigvee_{i\in[1,l]} \mathsf{BCMDecode}(pk, sk, c_i).$$

Let $c = (c_1, c_2) = (\{c_{1,j}\}_{j\in[0,n]}, (z, u)) = (\{\prod_{i=1}^{l} c_{i,1,j}^{\lambda_i}\}_{j\in[0,n]}, (\prod_{i=1}^{l} z_i^{\lambda_i}, \prod_{i=1}^{l} u_i^{\lambda_i}))$ be a coupon obtained by combining l valid coupons $c_i = (\{c_{i,1,j}\}_{j\in[0,n]}, (z_i, u_i))$, then we have:

$$\begin{aligned} & e(z, g_2) \cdot e(u, \hat{g}_2) \cdot \textstyle\prod_{j=0}^{n} e(c_{1,j}, hp_j) \\ &= e(\prod_{j=0}^{n}\prod_{i=1}^{l} c_{i,1,j}^{-\xi_j\cdot\lambda_i}, g_2) \cdot e(\prod_{j=0}^{n}\prod_{i=1}^{l} c_{i,1,j}^{-\rho_j\cdot\lambda_i}, \hat{g}_2) \cdot \prod_{j=0}^{n} e(\prod_{i=1}^{l} c_{i,1,j}^{\lambda_i}, g_2^{\xi_j}\hat{g}_2^{\rho_j}) = 1_{\mathbb{G}_T} \end{aligned}$$

3.2 Security

We prove that our extended BCM construction satisfies indistinguishability, unforgeability and untraceability properties. The indistinguishability model in [2, 3, 5] reflects that, given a valid coupon and oracle access to dummy and signal coupons, the adversary cannot tell whether it is dummy or signal. Our model extends it by considering coupons with signals on different options, in addition to dummy ones.

As in [18], the unforgeability model prevents an adversary from generating a valid signal coupon which could not be a combination of coupons already seen.

Untraceability is a new BCM security property that encompasses the specific issues encountered in voting applications over unreliable networks, where an attacker should be precluded from tracking and following voters based on the information that coupons store. Given two sets of valid coupons embedding identical signals and oracle access to any coupons, the adversary should not able to tell which set was used to generate the combined coupon.

Indistinguishability Model. In this model, the adversary is provided with a coupon generation oracle, giving it access to as many dummy and signal coupons it wants. Then, given a valid coupon, the adversary cannot tell whether its nature with a non-negligible advantage. The experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Indist}}$ from Fig. 2 defines the security experiment for indistinguishability.

Experiment $\mathsf{Exp}^{\text{Indist}}_{\mathcal{A}}$

$(pk, sk) \leftarrow \mathsf{BCMGen}(1^k)$
$(\vec{v}_0, \vec{v}_1) \leftarrow \mathcal{A}^{\mathcal{O}_{gen}(pk,sk)}(pk)$
$b \leftarrow \{0,1\}$
$c_b \leftarrow \mathsf{BCMCouponGen}(pk, sk, \vec{v}_b)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{gen}(pk,sk)}(pk, c_b)$

Experiment $\mathsf{Exp}^{\text{Unforg}}_{\mathcal{A}}$

$(pk, sk) \leftarrow \mathsf{BCMGen}(1^k)$
$c^* \leftarrow \mathcal{A}^{\mathcal{O}_{gen}(pk,sk)}(pk)$

Note that $\mathcal{A}$ is only allowed to access $\mathcal{O}_{gen}$ once for each single signal vector.

Experiment $\mathsf{Exp}^{\text{Untrac}}_{\mathcal{A}}$

$(pk, sk, l \in \mathbb{N}) \leftarrow \mathsf{BCMGen}(1^k)$
$(\{c_{0,i}\}_{i\in[1,l]}, \{c_{1,i}\}_{i\in[1,l]}) \leftarrow \mathcal{A}^{\mathcal{O}_{gen}(pk,sk)}(pk, l)$ s.t.
$\mathsf{BCMDecode}(pk, sk, \mathsf{BCMCombine}(pk, \{c_{j,i}\}_{i\in[1,l]}))$ is the same for $j \in \{0,1\}$
$b \leftarrow \{0,1\}$
$c_b \leftarrow \mathsf{BCMCombine}(pk, \{c_{b,i}\}_{i\in[1,l]})$
$b' \leftarrow \mathcal{A}(pk, c_b)$

Figure 2: Algorithms defining security experiments $\mathsf{Exp}^{\text{Indist}}_{\mathcal{A}}$ for indistinguishability, $\mathsf{Exp}^{\text{Unforg}}_{\mathcal{A}}$ for unforgeability and $\mathsf{Exp}^{\text{Untrac}}_{\mathcal{A}}$ for untraceability.

The adversary wins if $b = b'$ in the experiment. Our scheme is $(T, \varepsilon, q_1 + q_2)$-indistinguishable if there is no T-time adversary $\mathcal{A}$ that succeeds with advantage $|\Pr[b = b'] - 1/2| < \varepsilon$ after making up to $q_1 + q_2$ queries to the coupon generation oracle $\mathcal{O}_{gen}$.

Indistinguishability Proof Sketch. Indistinguishability follows directly from the n-MDDH assumption. We sketch a challenger $\mathcal{B}$ against n-MDDH using an adversary $\mathcal{A}$ against indistinguishability with advantage ε. $\mathcal{B}$ takes as input $g_1^{x_0}$ and two tuples of group elements $\{g_1^{x_j}\}_{1\leq j\leq n}$ and $\{g_1^{t_{0,j}}\}_{1\leq j\leq n}$, where $x_0, x_j \in_R \mathbb{Z}_p$ and $t_{0,j}$ is either equal to $x_0 x_j$ or to a random exponent. When $\mathcal{A}$ submits queries to the coupon generation oracle $\mathcal{O}_{gen}$, $\mathcal{B}$ uses the coupon generation algorithm with its first group element tuple as the secret key. When $\mathcal{A}$ submits two challenged signals, $\mathcal{B}$ uses $g_1^{x_0}$ and its second group element tuple to create the coupon using one of the two challenged signals. If the adversary $\mathcal{A}$ guesses the signal correctly, $\mathcal{B}$ outputs 1; otherwise, it outputs 0.

If the input to $\mathcal{B}$ comes from $\mathsf{MDH}_{\mathsf{n}}$ (meaning that $t_{0,j} = x_0 x_j$), the above simulation will be a perfect one of the security game, and the probability that $\mathcal{B}$ outputs 1 equals the probability that the adversary $\mathcal{A}$ guesses correctly in the security experiment. However, if the input to $\mathcal{B}$ comes from $\mathsf{MDH}^{\$}_{\mathsf{n}}$ (meaning that $t_{0,j}$ is a random exponent), the challenged coupon will be independent of the adversary's challenged signals, and the probability that $\mathcal{A}$ guesses correctly is $1/2$. It then follows that $Adv^{mddh_n}_{\mathbb{G}_1}(\mathcal{B}) = \varepsilon$.

Unforgeability Model. In this model, the adversary is given oracle access to dummy coupons and single signal coupons such that the latter are each requested only once. The adversary is not able to create a valid signal coupon that is not a combination of queried coupons with a non-negligible probability. The experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{Unforg}}$ from Fig. 2 defines the security model for unforgeability.

The adversary outputs a valid forged coupon c^* if and only if $\mathsf{BCMDecode}(pk, sk, c^*) = \vec{v}^* \neq \perp$ and $\vec{v}^* \notin \{\bigvee_{i \in S} \vec{v}_i; S \subseteq [1, n]\}$ (i.e. linear span). Our scheme is (T, ε, q)-existentially unforgeable if there is no T-time adversary $\mathcal{A}$ that succeeds with at least probability ε after making up to n queries to the oracle $\mathcal{O}_{gen}$.

Unforgeability Proof Sketch. Before going through the proof for unforgeability, one should notice that signatures (z, u) are part of coupons, hence an adversary asking for signatures is essentially asking for coupons.

Let $\mathcal{A}$ be an adversary that forges a signature with non-negligible advantage. Let $\mathcal{B}$ be the challenger that takes as input a DP instance $(g_2, \hat{g}_2) \in \mathbb{G}_2^2$ and expects $\mathcal{A}$ to find a non-trivial pair $(z, u) \in \mathbb{G}_1^2$ such that $e(z, g_2) \cdot e(u, \hat{g}_2) = 1_{\mathbb{G}_T}$. To do so, $\mathcal{B}$ runs the algorithm KeyGen (when running BCMGen) with random exponents $\xi_j, \rho_j \in_R \mathbb{Z}_p$ for $j \in [0, n]$. When $\mathcal{A}$ requests a signature on $(g_1^{\delta}, \{h_j^{\delta}\}_{j \in [1,n]})$, $\mathcal{B}$ replies by running the algorithm Sign (when running $\mathsf{BCMCouponGen}$). Finally, the adversary $\mathcal{A}$ outputs $(g_1^{\delta^*}, \{h_j^{\delta^*}\}_{j \in [1,n]})$ with signature elements $c_2^* = (z^*, u^*)$. The challenger $\mathcal{B}$ computes a signature on $\mathcal{A}$'s input as $c_2^\dagger = (z^\dagger, u^\dagger) = (g_1^{\delta^\dagger \cdot (-\xi_0)} \cdot \prod_{j=1}^n h_j^{\delta^\dagger \cdot (-\xi_j)}, g_1^{\delta^\dagger \cdot (-\rho_0)} \cdot \prod_{j=1}^n h_j^{\delta^\dagger \cdot (-\rho_j)})$. With overwhelming probability, the ratios $\frac{z^*}{z^\dagger}$ and $\frac{u^*}{u^\dagger}$ are a non-trivial solution to the DP problem. Indeed, any public key pk has exponentially many corresponding secret keys, and thus pk perfectly hides the exponents $\{\xi_j\}_{j \in [0,n]}$ and $\{\rho_j\}_{j \in [0,n]}$. In addition, for a given public key pk, elements $g_1^{\delta^*}$ and $\{h_j^{\delta^*}\}_{j \in [1,n]}$ have an exponential number of valid signatures while Sign algorithm's output is completely determined by the exponents $\{\xi_j\}_{j \in [0,n]}$ and $\{\rho_j\}_{j \in [0,n]}$.

Over the game between $\mathcal{B}$ and $\mathcal{A}$, the latter obtains signatures $\{c_{i,2} = (z_i, u_i)\}_{i \in [1,n]}$ on at most n linearly independent vectors. Hence, given $\{hp_j\}_{j \in [0,n]}$, $\mathcal{A}$ sees at most $2n + 1$ linear equations in $2n + 2$ unknown values. In $\mathcal{A}$'s view, since $(g_1^{\delta^*}, \{h_j^{\delta^*}\}_{j \in [1,n]})$ must be independent of previously signed vectors, predicting $z^\dagger$ is done only with probability $1/p$. Therefore, with overwhelming probability $1 - \frac{1}{p}$, $z^* \neq z^\dagger$, and thus $\frac{z^*}{z^\dagger}$ and $\frac{u^*}{u^\dagger}$ is a valid solution to the DP problem.

Untraceability Model. In this model, the adversary is provided with a coupon generation oracle, giving it access to as many dummy and signal coupons it wants, such that two distinct sets of these coupons decode to the

same vector. Then, given a valid combined coupon, the adversary cannot tell from which set it results from with a non-negligible advantage. The experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Untrac}}$ from Fig. 2 defines the security model for untraceability.

The adversary wins if $b = b'$. Our scheme is (T, ε, q)-untraceable if there is no T-time adversary $\mathcal{A}$ that succeeds with advantage $|\Pr[b = b'] - 1/2| < \varepsilon$ after making up to q queries to the coupon generation oracle $\mathcal{O}_{gen}$.

Untraceability Proof. Before going through the proof for untraceability, one should notice that Left-or-Right and Real-or-Random notions are equivalent. Since the former is more natural as a security notion, we use it as our security definition. Since the later is easier to work with, we use it as our proof.

Let an adversary $\mathcal{A}$ against untraceability with advantage ε. Using $\mathcal{A}$, we build a challenger $\mathcal{B}$ that solves the n-DDHE problem in $\mathbb{G}_1$. For a generator $g_1 \in \mathbb{G}_1$ and an exponent $\nu \in \mathbb{Z}_p^*$, let $y_j = g_1^{\nu^j} \in \mathbb{G}_1$ for $j \in [1, n]$. The challenger $\mathcal{B}$ is given as input a tuple $(X, T) = ((g_1, y_1, \cdots, y_n), T)$ where T is either equal to $y_{n+1} = g_1^{\nu^{n+1}}$ or to a random element in $\mathbb{G}_1^*$ (such that T is uniform and independent in $\mathbb{G}_1^*$). The challenger's goal is to output 0 when $T = y_{n+1}$ and 1 otherwise. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Let g_2 be a generator of $\mathbb{G}_2$, $\alpha \in_R \mathbb{Z}_p$ and $\hat{g}_2 = g_2^{\alpha}$. Let $\xi_j, \rho_j \in_R \mathbb{Z}_p$, and $hp_j = g_2^{\xi_j} \hat{g}_2^{\rho_j}$ for $j \in [0, n]$. The challenger sets $h_j = y_j$ for $j \in [1, n]$, meaning that the exponents β_j are implicitly equal to ν^j (and hence, the elements β_j remain unknown to $\mathcal{B}$). The public key $pk = (g_1, g_2, \hat{g}_2, \{hp_j\}_{j\in[0,n]})$ is provided to $\mathcal{A}$. The secret key $sk = (\{\xi_j, \rho_j\}_{j\in[0,n]}, \{h_j\}_{j\in[1,n]})$ is kept by $\mathcal{B}$. The adversary makes queries on vector $\vec{v}$ to the challenger. The latter replies by generating coupons as follows. First, $\mathcal{B}$ chooses at random exponents $\delta_0, \delta_1, \cdots, \delta_n \in_R \mathbb{Z}_p$. Two or more exponents are equal when there is no signal, while unique exponents represent signals. Then, it computes $c_{1,0} = g_1^{\delta_0}$, $c_{1,j} = y_j^{\delta_j}$ for $j \in [1, n]$, $z = g_1^{-\xi_0 \cdot \delta_0} \cdot \prod_{j=1}^{n} y_j^{-\xi_j \cdot \delta_j}$ and $u = g_1^{-\rho_0 \cdot \delta_0} \cdot \prod_{j=1}^{n} y_j^{-\rho_j \cdot \delta_j}$. Such coupons are computable from elements in the tuple given to the challenger.

After the query phase, $\mathcal{A}$ submits two challenged coupon sets $\{c_{0,i}\}_{i\in[1,l]}$ and $\{c_{1,i}\}_{i\in[1,l]}$ such that both sets contain the same signals (if any); and both sets contain the same number of coupons (note that we can pad one set with dummy coupons if needed). $\mathcal{B}$ picks at random a bit $b \in_R \{0, 1\}$. For $i \in [1, l]$, we denote a coupon $c_{b,i}$ as $(c_{1,i}^{(b)}, c_{2,i}^{(b)}) = (\{c_{1,j,i}^{(b)}\}_{j\in[0,n]}, (z_i^{(b)}, u_i^{(b)}))$. The challenger computes the coupon c_b that combines $c_{b,i}$ for $i \in [1, l]$ as follows: $c_{1,0}^{(b)} = \prod_{i=1}^{l} y_1^{\delta_{0,i}^{(b)}} = \prod_{i=1}^{l} (g_1^{\nu})^{\delta_{0,i}^{(b)}} = \prod_{i=1}^{l} (c_{1,0,i}^{(b)})^{\nu}$, $c_{1,j}^{(b)} = \prod_{i=1}^{l} y_{j+1}^{\delta_{j,i}^{(b)}} = \prod_{i=1}^{l} (y_j^{\nu})^{\delta_{j,i}^{(b)}} = \prod_{i=1}^{l} (c_{1,j,i}^{(b)})^{\nu}$ for $j \in [1, n-1]$ and $c_{1,n}^{(b)} = \prod_{i=1}^{l} T^{\delta_{n,i}^{(b)}}$. If $T = y_{n+1}$ then $c_{1,n}^{(b)} = \prod_{i=1}^{l} T^{\delta_{n,i}^{(b)}} = \prod_{i=1}^{l} (y_n^{\nu})^{\delta_{n,i}^{(b)}} = \prod_{i=1}^{l} (c_{1,n,i}^{(b)})^{\nu}$, meaning that the coupon c_b is valid. Indeed, the exponents $\lambda_i \in \mathbb{Z}_p$ are all implicitly

set to be equal to ν, and correctness on combined coupon decoding follows. Otherwise, the coupon c_b is independent of b in $\mathcal{A}$'s view.

Finally, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. The challenger concludes its own game by outputting a guess as follows. If $b = b'$, then $\mathcal{B}$ outputs 0, meaning that $T = y_{n+1} = g_1^{\nu^{n+1}}$; otherwise, it outputs 1, meaning that T is random in $\mathbb{G}_1$. When $T = y_{n+1} = g_1^{\nu^{n+1}}$, then $\mathcal{A}$'s view is identical to its view in a real attack game, hence $|\Pr[b = b'] - 1/2| \geq \varepsilon$. When T is random in $\mathbb{G}_1$, then $\Pr[b = b'] = 1/2$. Therefore, for g_1 and T uniform in $\mathbb{G}_1$, for ν uniform in $\mathbb{Z}_p$, $|\Pr[\mathcal{B}(X, y_{n+1}) = 0] - \Pr[\mathcal{B}(X, T) = 0]| \geq |(1/2 \pm \varepsilon) - 1/2| = \varepsilon$.

3.3 Efficiency Analysis

Table 1 shows the efficiency results of our extended BCM scheme. The size of components (public key, secret key, coupon) is linear in the number of options, i.e. $O(n)$. The process of checking the validity of a coupon requires $O(n)$ pairing operations, since every element of a coupon is carefully included in the verification. By doing so, any (malicious) modification on a coupon will be notified with a verification failure. The process of decoding a coupon involves $O(n)$ exponentiation operations, since the first $n + 1$ elements of a coupon (excluding the signature part) are required to interpret the veto outcome, which must likely be correct.

Public key pk (# elements)	Secret key sk (# elements)	Coupon c (# elements)	Verification time (# pairings)	Decoding time (# exponentiations)
$n + 4$	$4n + 2$	$n + 3$	$n + 3$	n

Table 1: Size of public/secret keys and coupons, and verification and decoding costs for coupons with n options.

The beneficial feature enabling secure voting over unreliable networks inevitably has an effect on the efficiency of our solution. Yet, while our voting protocol is not optimally efficient, it is practical and can be run in most, if not all, circumstances that require veto elections.

A cost that is linear in the number of vote options is normal in voting systems – for instance, when using homomorphic voting and with a 1-out-of-n scheme involving n encryptions. Our scheme is in some sense a 1-out-of-$\binom{n'}{n}$ scheme for $n' \in [0, n]$ when voters choose to veto on n' options. This means that the ciphertext needs to be of size $O(n)$, and thus the cost of encryption must be at least $O(n)$. Even with such cost, the scheme is practical since it can easily be implemented using terminals, such as reasonably modern smartphones. Clearly, there are more efficient schemes, but they do not have the properties we want.

A parallelization of the protocol may at first seem a promising trade-off between efficiency and security for some scenarios. Indeed, transmitting

signals in a parallel way allows voters to only spread coupons containing one signal and no dummies, and thus being of size not depending on the value n. However, in a parallel version of BCM, a coupon with a signal on one option should contain information about which option is considered, increasing the size of the coupon. In addition, a parallel fashion may lose vetoes: coupons with one specific option may be dropped more often, and thus a signal on that option would less likely be noticed. The frequency of spreading coupons containing signals on that particular option may also be higher, and thus noticeable, potentially leaking information to unauthorized entities. In other words, the parallel version does not achieve the desired security goals. On the other hand, transmitting whole coupons with $O(n)$ elements allows to increase the chances to intercept the signals embedded in these coupons, while avoiding leaking information from the frequency of signals on given options.

4 Application: Veto Voting Protocol

The BCM is a primitive originally designed to spread alerts quietly and quickly on wireless networks [2, 3, 5]. We propose a new application from such primitive where our multivariate extension makes it beneficial: peer-to-peer veto voting over unreliable networks. In this section, we describe a protocol where m election authorities are responsible to prepare the veto election and release the veto results, and manifold voters make their choices on n decisions and interact each other during the voting session.

4.1 Description

Setup Phase. The setup phase is managed by m election authorities. First, the election authorities run the algorithm BCMGen in a distributed way to generate the public and secret parameters. To proceed, we assume that election authorities use existing tools such as secret sharing and multi-party computation [19, 20, 12, 10] (this process is out of scope of our paper). Public parameters are made available to all participants, while secret parameters are shared among the m election authorities. Second, the election authorities run the algorithm $\mathsf{BCMCouponGen}$ in a distributed way to generate n single veto coupons (corresponding to n single signal coupons) and one blank coupon (corresponding to a dummy coupon) for each voter over secure communication channels.

Every voter (assigned with an index i) hence receives $n+1$ coupons $c_{i,0}, c_{i,1}, \cdots, c_{i,n}$. There are n options that voters can veto on, where each option refers to the cessation of a demonstration for instance. The coupon $c_{i,0}$ corresponds to a blank vote (no veto) and a coupon $c_{i,j}$, for $j \in [1,n]$, corresponds to a veto on option opt_j. Each voter can combine her coupons to obtain multiple veto option coupons. For instance, by combining the

coupon $c_{i,1}$, that corresponds to a veto on option opt_1, and the coupon $c_{i,2}$, that corresponds to a veto on option opt_2, the resulting coupon c is for vetoes on both options opt_1 and opt_2. Once all voters get the public key material and their coupons, the setup phase is over. From this point onwards, secure communication channels between the authorities and voters is not assumed.

Voting Phase. A voter V_A starts the voting phase by generating and spreading blank coupons to her neighbors. We suppose such action from the voter since we want to create a continuous spread of dummy and signal coupons over the voting phase, such that attackers only see the uninterrupted flow while not distinguishing the nature of these coupons.

Thereafter, the voter V_A may decide to veto on either a single option or multiple options. To do so, she distributes a coupon $c1$ that either belongs to $\{c_{i,j}\}_{j\in[1,n]}$ (for vetoing on a single option) or is a combination of single veto coupons from $\{c_{i,j}\}_{j\in[1,n]}$ (for vetoing on multiple options). Combination of coupons is enabled by running the algorithm BCMCombine. Let V_B be one of V_A's neighbors that has received $c1$. At the moment of reception, V_B holds a coupon $c2$ (representing either a blank vote, a single veto vote or a multiple veto vote). First, V_B checks the validity of the coupon $c1$ by running the algorithm BCMVerify. If the output is `valid`, then he runs the algorithm BCMCombine to combine $c1$ and $c2$ resulting into $c3$, and spreads the latter to his neighbors.

The process of generating and spreading coupons is performed by all voters in the voting network. Each election authority is enabled to communicate with voters individually, allowing the former to intercept emitted coupons and verify with other authorities whether vetoes have been triggered, and on which option(s). The voting phase is over after a certain time that has been agreed among the m election authorities.

Note that a dishonest voter cannot forge a coupon, meaning that no coupon can be created that was not received from the authorities. Therefore a dishonest voter V_A can only deviate from the correct protocol by either spreading coupons with vetoes on additional options, or by not combining received coupons and only transmitting dummy coupons. In the former case, then V_A just expresses her opinion, that is, she is against the chosen propositions. In the second case, then V_A acts as she agrees on all propositions, and other (honest) voters combine her dummy coupons with their own ones, hence possibly inserting some vetoes. We assume that not all the voters act like that since this is in their interest to spread vetoes if they want to raise their voices (anonymously) against their government for example. Therefore, some legitimate coupons are spread anyway and likely intercepted by the authorities.

Sampling Phase. The m election authorities jointly decode the coupons intercepted so far by distributively running the algorithm BCMDecode and obtain the veto results. More precisely, elements in a decoded vector tell whether vetoes have been launched, and on which option(s). Election authorities can repeat the sampling process on multiple coupons in order to enhance the veto results. Depending on the number of collected coupons and the time allocated for voting, the election authorities can attest with overwhelming probability the results of the veto election. Election authorities do not count the number of vetoes since absolute veto is considered, meaning that one veto is enough to stop an action.

We observe that no strong assumption on the reliability of communication channels between the authorities and voters is required. Communication channels are supposed to be functional enough to allow at least one authority to receive one coupon, at the expense of a lower probability on the accuracy of the election result.

A malicious voter may have never combined her neighbors' coupons and only forwarded coupons embedding her own non-veto and veto choices only. Hence, if all voters have acted in such way, then care needs to be taken by the election authorities when sampling. Combination of intercepted coupons can be done by the latter, preventing biased results from dishonest voters. We can also assume that at least one voter is honest and that coupons from this voter has been received and decoded by the election authorities.

4.2 Security

We discuss in this section the security requirements [11] that our veto voting system should meet. Security requirements of the voting protocol reduce to the security of the extended BCM, that has been carefully proven secure in Section 3.2. Due to lack of space, we opt for a succinct security description of the resulting veto protocol.

Privacy. For each election, votes should be as secret as possible. An exception exists when a final result only contains non-veto votes, thus no vote can be private. We do not consider such case when discussing about privacy. Privacy comes from indistinguishability: since an attacker cannot distinguish coupons, no information about (non-)veto is leaked. Privacy also derives from untraceability: an attacker cannot find out the path taken by a coupon, assuring that the voting choices made by voters cannot be successfully guessed.

Anonymity and Fairness. Anonymity implies that no one can discover which voter vetoed which option(s). Indeed, an attacker must not be able to reveal the identity of a voter from an intercepted coupon. This requirement is guaranteed by untraceability: an attacker cannot find the path followed

by a coupon, and thus cannot locate the voter who emitted the coupon. One can argue that anonymity can lead to potential misuse of the veto power and some kind of accountability on the voter who casts a veto can be desirable. However, in our scenario, we believe anonymity is more important than accountability.

A fairness guarantee should ensure that a given voting method is sensitive to all of the voters' opinions in the right way. Fairness implies that no early results from voting can be obtained. Indeed, none of the voters can learn information on the election outcome before the election authorities officially reveal it. From receiving coupons from neighbors, voters are not able to know whether vetoes are embedded, since any two coupons are indinstinguishable and secret key material is required to decode them. This also applies for any external spotters intercepting coupons during the voting phase. Election authorities may intercept coupons during the voting phase and jointly decode them to get a partial result. Nevertheless, we assume that enough election authorities (up to a threshold defined according to the underlying distribution technique [19, 20, 12, 10]) wait for the sampling phase to decode coupons and recover the election results.

Partial Verifiability. Each voter should be able to check that tallying and counting were performed correctly. Unfortunately, our solution does not permit a voter to be aware that her vote has been correctly taken into consideration in the final result. If the voter has vetoed on option opt_j, and if the decoded vector contains a veto for opt_j, then the voter is relieved that her veto choice is included in the election result but cannot be guaranteed that her veto has been assessed. We recall that vote counting is not necessary for absolute veto elections. If the voter has vetoed on option opt_j, and if the decoded vector does not contain a veto for opt_j, then the voter is aware that her veto vote has not been taken into account. In order to increase the probability of having her vote included in the final result, the voter transmits multiple times her veto coupon to her neighbors. We aim to develop a substantive, efficient veto protocol in unreliable communication infrastructures. While verifiability is desirable, it does not seem essential in our scenario.

Correctness and Robustness. Correctness implies that no one should be able to submit incorrect votes. Correctness is enhanced with the algorithm BCMVerify that permits everyone to check the validity of a coupon. Correctness also originates from unforgeability, guaranteeing that an attacker cannot submit a valid fake coupon that successfully decodes to a consistent vote.

Robustness implies that a malicious voter should not prevent the election result from being declared. Even if some voters either fail to vote or abort the

voting phase, the election result can still be announced. Election authorities can collect coupons at the time of abortion, and decode them to know what has been vetoed so far.

Functionality. Our voting protocol requires a new setup phase as soon as veto options change. However, if several elections contain the same options, then there is no need to generate and deliver new coupons. Voting and sampling phases should be executed for each election. Nevertheless, both phases can be parallelized: election authorities jointly decode coupons during the voting phase by intercepting them, and agree on a final result once this result is obtained with high probability from collected coupons.

Partial Collusion Resistance. A full collusion against one specific voter involves all other voters in the network. An anonymous protocol is by definition not fully collusion resistant [9, 14] since the voter's anonymity cannot be preserved. Nevertheless, having all voters acting maliciously and colluding against one particular voter is not possible in practice; otherwise, this voter would just decide to leave the network. Our voting protocol is partially resistant against colluding voters, such that they cannot discover the result of the veto election and cannot force a result more that what they have been given through their coupons. Such guarantees come from indistinguishability and unforgeability. Our protocol is also resistant against colluding election authorities, up to a given threshold. Such assurance is determined by the distribution technique used at the setup phase [19, 20, 12, 10].

No Self-Tallying. The voting protocol is not self-tallying since election authorities must use their secret key shares to decode coupons.

5 Observations and Future Work

We discuss here some observations from both our BCM scheme and veto protocol and possible improvements that can be brought.

Enhancing Unforgeability. Future work will focus on designing a BCM solution that guarantees a stronger unforgeability level while saving its applicability in constrained peer-to-peer communication infrastructures. Our current model restricts the adversary to submit unique coupons with one signal option to the oracle. A desirable extension would be to allow the adversary to request coupons with multiple signal options.

Enhancing Untraceability. Veto elections within a dynamic constrained network should offer untraceability guarantees that would prevent malicious voters and election authorities from tracking, following and distinguishing

coupons. We have proven our solution untraceable based on a model where the adversary is only given the public key material and access to a coupon generation oracle. Such model can be strengthened by giving the adversary either the secret key material or access to a coupon decoding oracle. In the full version of the paper, we present three stronger untraceability models and discuss the benefits and limitations of such models regarding our solution and the ones from [2, 3, 5].

Accountability. For our scenario, we believe anonymity is more important than accountability. However, it is possible to add some accountability to the system. For instance, we can design a threshold system of accountability, such that if fewer voters than the threshold number submit vetoes, then they will be identifiable by someone with the appropriate secret key. On the other hand, if more voters than the threshold number submit vetoes, then they cannot be identified. We discuss this further in the full version.

Unauthorized Voters. Election authorities can forbid some voters to veto by giving them $n+1$ dummy coupons. Since dummy and signal coupons are indinstinguishable, these voters are not aware of being unauthorized to veto. In our scenario, this setting enables the organizers to specifically authorize experienced demonstrators to decide for demonstration cessation, while inexperienced ones cannot. To go easier on their egos, the latter are not told directly that they have no veto competence. We stress that this is entirely optional.

6 Conclusion

In this paper, we proposed an extension of Blazy-Chevalier BCM scheme [5] to enable the quiet propagation of multiple signals. Our BCM solution is proved secure with relation to indistinguishability and unforgeability properties. We also defined a new security notion, that is untraceability, and showed that our BCM construction satisfies it. Finally, based on our enhanced BCM, we presented an efficient and reliable veto protocol using peer-to-peer communications over unreliable networks.

Acknowledgments

This work was supported by the the Luxembourg National Research Fund and the Research Council of Norway for the joint project SURCVS.

References

[1] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. *J. Cryptol.*, 29(2):363–421, Apr. 2016.

[2] J. Aspnes, Z. Diamadi, K. Gjøsteen, R. Peralta, and A. Yampolskiy. Spreading alerts quietly and the subgroup escape problem. In *Proc. of the 11th International Conference on Advances in Cryptology*, ASIACRYPT'05, pages 253–272. Springer, 2005.

[3] J. Aspnes, Z. Diamadi, A. Yampolskiy, K. Gjøsteen, and R. Peralta. Spreading alerts quietly and the subgroup escape problem. *J. Cryptol.*, 28(4):796–819, Oct. 2015.

[4] S. Bag, M. A. Azad, and F. Hao. Priveto: a fully private two-round veto protocol. *IET Information Security*, 13(4):311–320, July 2018.

[5] O. Blazy and C. Chevalier. Spreading alerts quietly: New insights from theory and practice. In *Proc. of the 13th International Conference on Availability, Reliability and Security*, ARES'18, pages 30:1–30:6. ACM, 2018.

[6] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proc. of the 25th International Conference on Advances in Cryptology*, CRYPTO'05, pages 258–275. Springer-Verlag, 2005.

[7] F. Brandt. Efficient cryptographic protocol design based on distributed El Gamal encryption. In *Proc. of the 8th International Conference on Information Security and Cryptology*, ICISC'05, pages 32–47. Springer-Verlag, 2006.

[8] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In *Proc. of the International Conference on Advances in Cryptology*, EUROCRYPT'02, pages 321–336. Springer-Verlag, 2002.

[9] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, Mar. 1988.

[10] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. of the 28th Annual Symposium on Foundations of Computer Science*, SFCS'87, pages 427–438. IEEE Computer Society, 1987.

[11] K. Gjøsteen. A latency-free election scheme. In *Proc. of the International Conference on Topics in Cryptology*, CT-RSA'08, pages 425–436. Springer Berlin Heidelberg, 2008.

[12] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th Annual Symposium on Theory of Computing*, STOC'87, pages 218–229. ACM, 1987.

[13] J. Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *Proc. of the International Conference on Financial Cryptography*, FC'04, pages 90–104. Springer Berlin Heidelberg, 2004.

[14] F. Hao and P. Zieliński. A 2-round anonymous veto protocol. In *Proc. of the International Workshop on Security Protocols*, Security Protocols'06, pages 202–211. Springer Berlin Heidelberg, 2006.

[15] J. Herranz, F. Laguillaumie, B. Libert, and C. Rafols. Short attribute-based signatures for threshold predicates. In *Proc. of the 12th Conference on Topics in Cryptology*, CT-RSA'12, pages 51–67. Springer-Verlag, 2012.

[16] D. Khader, B. Smyth, P. Y. A. Ryan, and F. Hao. A fair and robust voting system by broadcast. In *Proc. of the 5th International Conference on Electronic Voting*, EVOTE'12, pages 285–299. Gesellschaft für Informatik, 2012.

[17] A. Kiayias and M. Yung. Non-interactive zero-sharing with applications to private distributed decision making. In *Proc. of the International Conference on Financial Cryptography*, FC'03, pages 303–320. Springer Berlin Heidelberg, 2003.

[18] B. Libert, T. Peters, M. Joye, and M. Yung. Linearly homomorphic structure-preserving signatures and their applications. *Des. Codes Cryptography*, 77(2-3):441–477, Dec. 2015.

[19] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[20] A. C. Yao. Protocols for secure computations. In *Proc. of the 23rd Symposium on Foundations of Computer Science*, SFCS'82, pages 160–164, 1982.