Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols

Josh Cohen Benaloh*

1 Introduction

This paper describes a deceptively (almost embarrassingly) simple technique, that of *cryptographic capsules*, which allows Alice to convince Bob that either X or Y is true without giving Bob any information as to which is the case. Capsules are an instrumental part of the machinery used to compose ballots in the cryptographic election scheme of [CoFi85] (see also [Coh86], [Ben86], and [BeYu86]), but they have far broader applications. Use of capsules substantially simplifies the "zero-knowledge" interactive proof system for quadratic non-residuosity published in [GMR85]. Their use also provides a tremendous simplification of the "result-indistinguishable" interactive proof system published in [GHY85]. Capsules have been incorporated into the zero-knowledge protocol for interactively proving non-isomorphism of graphs described in [GMW86]. Finally, capsules are shown here to provide a mechanism more efficient than that of [GMW86] by which Alice can convince Bob (in a zero-knowledge fashion) of the validity of *any* $\mathcal{NP}$ predicate.

Despite their simplicity, it seems that the applications of capsules may go far beyond those mentioned here, and capsules have the potential to become a standard primative construct for many kinds of interactive protocols.

2 Cryptographic Capsules

A *cryptographic capsule* (or simply *capsule*) is a randomly ordered collection of objects, each of which is of some specified form. The order of the elements of the capsule is randomly permuted to hide which element is of which type; or, alternately, some easily computable ordering function (such as $\leq$) can be applied to the capsule to obscure the original ordering.

*This work was supported in part by the National Security Agency under Grant MDA904-84-H-0004.

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 213-222, 1987.

A simple example of a capsule is a pair of integers — one of which is even and one of which is odd, e.g. $(4, 13)$. This capsule, however, is not very interesting because it is readily apparent *which* is the odd integer and which is the even integer.

A somewhat more useful capsule may be an (unordered) pair of integers $\{n_1, n_2\}$ with $n_1 = p_1 q_1$ where p_1 and q_1 are each primes congruent to 1 modulo 4 and $n_2 = p_2 q_2$ where p_2 and q_2 are each primes congruent to 3 modulo 4.

If we assume that distinguishing between these two cases is hard, then this suggests a simple method for flipping a coin over a telephone. Alice prepares such a pair and transmits it to Bob; Bob then selects one element from the pair and transmits his choice to Alice; finally, Alice reveals the factors of both n_1 and n_2 to Bob. We may say that the coin flip is heads if Bob chose the element with factors congruent to 1 modulo 4 and tails otherwise.

This is not an ideal example, since Alice could have simply transmitted a single integer of one of the two preceding classes and waited for Bob to guess which class it was from. The real power of capsules comes from the ability to prove interactively that a capsule is of the required form without the need to later reveal secret information about its contents.

3 Residue Classes and Capsules

Most of the interesting applications of cryptographic capsules so far explored involve their use with residue classes. The feature of residue classes which is important for this application is that two integers can be shown to be of the same residue class without giving any information about the actual residue classes to which the integers belong.

Formally, for any given integers n and y, y is said to be an r^{th} *residue* modulo n if and only if there exists some integer x such that $y \equiv x^r \pmod{n}$. The following lemma characterizes residue classes.

Lemma 1 *Let $\varphi(n)$ denote the Euler totient function, and choose n and r such that $r|\varphi(n)$ and $r^2 \not| \varphi(n)$. If y is relatively prime to n and is* not *an r^{th} residue modulo n, then every w which is relatively prime to n is expressible as $w \equiv x^r y^i \pmod{n}$ for a unique integer i in the range $0 \leq i < r$.*

This i is the *residue class* of w with respect to n, y, and r.

An important (although slightly variant) special case occurs when $r = 2$, and n is the product of two distinct primes. We ignore the choice of y here and denote the set of quadratic residues by class 0 and the set of quadratic non-residues with Jacobi symbol 1 by class 1.

A property of residue classes is apparent from the definition.

Lemma 2 *If x_1 and x_2 are members of residue classes i_1 and i_2, respectively, then the product x_1x_2 is a member of residue class $i_1 + i_2$.*

Note that for all integers i, residue classes i and $i + r$ are different denotations of the same class. The canonical denotation of a residue class I will be the unique class i with $0 \leq i < r$ such that $i \equiv I \pmod{r}$.

Finally, the following lemma shows how two integers can be shown to be of the same residue class.

Lemma 3 *Two integers x_1 and x_2 which are relatively prime to n are of the same residue class with respect to n, y, and r if and only if there exists some integer v such that $v^r \equiv x_1/x_2 \pmod{n}$.*

Thus, to prove that two integers are of the same residue class, it is necessary only to exhibit an r^{th} root of their quotient.

4 Some Applications

4.1 Elections

In the cryptographic election work of [CoFi85], each voter prepares, as a ballot, a capsule which consists of of a random member of residue class 0 (denoting a **no** vote) and a random member of residue class 1 (denoting a **yes** vote). Later, each voter will designate one of the components of his or her capsule as the actual vote. The votes can then be multiplied together, and (by Lemma 2) the resulting product is a member of residue class t, where t is the total number of **yes** votes. A powerful agent (such as a government) which holds the factorization of the modulus n used can then prove to all participants that the computed product is of residue class t *without* giving any additional information about the residue classes of the factors, thus protecting the privacy of the individual votes.

Where do capsules come in? It is essential that the vote cast by each voter be a member of either class 0 or class 1. If a voter were, for example, able to cast a vote of class 1,000,000, then this one vote would increment the tally by 1,000,000. The voter, however, does not want to reveal to which of class 0 or class 1 his or her vote belongs.

To prove that a chosen capsule C is of the required form, a voter engages in an interactive proof (see [FMR84] and [GMR85]). Each voter prepares a set B of (say) 100 additional capsules — each one, as the original, consisting of a random member of residue class 0 and a random member of residue class 1. Random bits are then generated[1], and used to partition B into sets S and T. The capsules of

[1] We assume here that some generally trusted source of randomness can be obtained, perhaps by *XOR*ing random bits generated by all (or some trusted subset) of the participants. In the other protocols described, the number of agents is small (usually two), and the challenging agent can generate its own random numbers.

set S are all "opened" to prove that they each consist of a proper **no** vote and a proper **yes** vote. (To open a capsule, a voter "opens" each component w of the capsule by revealing integers x and i, $i \in \{0,1\}$, such that $w \equiv x^r y^i \pmod{n}$ — see Lemma 1.) Each capsule in T is shown to be "equivalent" to C by showing that it has one component of the same class as the first component of C and one component of the same class as the second component of C. (Recall that by Lemma 3, two integers can be shown to be of the same residue class by showing that there quotient q is an r^{th} residue, and this in turn can be shown by exhibiting an r^{th} root of q.)

Once this process has been completed, it is known that every capsule in S is of the required form (one integer of class 0 and one integer of class 1), and every capsule of T is of the same form as C. Thus, C is of the required form unless *every* capsule in T is improper. Since the partition of B into S and T was chosen randomly *after* the capsules of B were prepared, C could only be improper if the partition were somehow guessed in advance. But the probability of doing this successfully is only 1 in 2^{100}. Hence, there is extremely high confidence that C is a proper capsule, and the voter can then vote by selecting one of the components of C.

Formal proofs that this procedure does not yield any extraneous information are included in [CoFi85].

4.2 Quadratic Residuosity

The work on elections has been previously published (besides [CoFi85], see [Coh86], [Ben86], and [BeYu86] for some extensions), and the above sketch is included only to motivate the use of capsules. In section 4.2, we shall examine how the use of capsules can greatly simplify protocols which have been published in [GMR85] and [GHY85].

4.2.1 Zero-Knowledge Non-residuosity

In [GMR85], a protocol is given whereby Alice convinces Bob that a given y is *not* a quadratic residue modulo a given n. (It is presumed that Alice has the factorization of n and that Bob does not.) Alice convinces Bob that y is not a residue by demonstrating her ability to distinguish members of a set X of randomly chosen residues from members of a set Y consisting of elements formed by multiplying other randomly chosen residues by y. If y were a residue, then all of the elements of X and Y would be random residues (class 0), and Alice would have no hope of distinguishing between them with better than a 50% chance. If, however, y is not a residue, then the elements of Y would be random elements of class 1. With the factorization of n, Alice can distinguish between elements of class 0 and elements of class 1 flawlessly.

In order to avoid acting as a residuosity oracle for Bob, Alice wants to be certain that the numbers she distinguishes between are generated by the protocol, i.e. before telling Bob whether a w which he has produced is a residue or a non-residue, Alice wants to be certain that Bob already *knows* which is the case (under the assumption that y is not a residue). To accomplish this, the authors include a rather cumbersome protocol in which Bob prepares (say) 100 elements of both types, "opens" those designated by Alice, opens additional elements to balance the types remaining, and applies one of four functions to w and each remaining element v according to the classes of w and v.

The simple process of grouping the elements into capsules eliminates the need for the balancing and the four separate functions (as well as the accompanying analyses). The process is essentially the same as a one voter election (Bob is the voter and Alice is the government).

Bob sends Alice a w generated either as a residue or as a product of a residue and y. Bob then prepares and sends to Alice (say) 100 capsules, each of which consists of a randomly chosen residue and the product of y and another random residue. Alice then randomly decides for each capsule whether or not it is to be opened. Those capsules designated by Alice are opened by Bob proving that they are of the stated form. From each remaining capsule, Bob chooses one element, which shall be denoted by x, and shows that x is of the same class as w by revealing a root of the quotient x/w — this demonstrates that if Bob can determine the class of x, he can also determine the class of w since they are the same by Lemma 3. As before, unless Bob already has sufficient information to determine the class of w without Alice's help, Bob has only 1 chance in 2^{100} of successfully answering Alice's challenges.

4.2.2 Result-indistinguishable Residuosity

[GHY85] generalizes the result of [GMR85] in such a way that an observer, Carol, watching the protocol between Alice and Bob gains no information from the protocol as to whether Alice convinced Bob that a given z was or was not a quadratic residue.

The key addition to the protocol of [GMR85] is the inclusion of a third set of possibilities. Instead of choosing w from among just two sets X and Y, Bob may select from an additional set Z. Members of X are randomly generated residues (class 0); members of Y are randomly generated non-residues (class 1) — these can be produced by multiplying random residues by a known non-residue y; finally, members of Z are generated by multiplying random residues by z (all elements of Z are of the same class as z).

To prove to Alice that she is not providing Bob with too much information, Bob must send Alice the (scrambled) members of 4 sets (essentially of the form of X, Y, Z, and $\overline{Z}$ — a complementary set to Z needed to maintain symmetry).

The remainder of the protocol is similar to [GMR85], except that the unrevealed portions of four sets instead of just two have to be simultaneously balanced (necessitating an even more arduous analysis), and a four by three table of functions is needed corresponding to which set w is a member of and which class each unopened element is a member of.

By using three-component capsules, the protocol of [GHY85] can be simplified tremendously. Bob simply prepares a master capsule C, cosisting of one member of each of X, Y, and Z, and (say) 100 additional scratch capsules of the same form. Alice designates some subset of the scratch capsules, and Bob opens these. Bob then shows that each remaining scratch capsule is equivalent to C by matching components and showing that their quotients are residues. Alice (now convinced that C was generated as required) tells Bob which capsule component is of a class *different* from the other two — thus transmitting to Bob the class of z.

The chance of Alice being fooled into revealing excessive information to Bob is only 1 in 2^{100}. The chance of Alice fooling Bob in one iteration of this protocol is 1/2, so by iterating the process, Bob can obtain extrememly (exponentially) high confidence that he has not been misled. Finally, it is not hard to show that Carol receives absolutely no information from watching this protocol that she could not have obtained on her own.

The necessary proofs of both [GMR85] and [GHY85] remain unchanged except for some straightforward simplifications and the removal of some analyses which are no longer necessary when the revised protocols are used.

4.3 Graph Non-isomorphism

One example in which capsules are useful *without* the aid of residue classes is seen in a protocol for graph non-isomorphism described in [GMW86]. Their original protocol closely followed the non-residuosity protocol of [GMR85]. Here, a prover designates a graph H given by the verifier as either a permutation of graph G_1 or of graph G_2 only after being convinced that the prover already holds such a permutation. Their protocol now incorporates capsules in a manner similar that described in Section 4.2.1 (residue classes are replaced by the equivalence classes induced by graph isomorphism, and class equivalence is demonstrated by exhibiting permutations). With this modificiation, their protocol and its analysis have been simplified.

5 Boolean Circuit Satisfiability

Very recently (also in [GMW86]), Goldreich, Micali, and Wigderson gave a simple and elegant zero-knowledge interactive protocol to prove for any k that a graph is k-colorable *without* revealing any information about a specific coloring (note that it is assumed that the prover possesses a k-coloring of the graph). Because

k-colorability is $\mathcal{NP}$-complete, this means that *any* positive instance of a problem in $\mathcal{NP}$ for which a prover holds a certificate (e.g. a satisfying assignment for a Boolean formula) can be reduced to graph colorability and shown in a zero-knowledge fashion to be a positive instance. The only assumption made is the existence of a probabilistic cryptosystem which is implied by the existence of a one-way permutation ([GoMi84],[Yao82]).

In this section, we shall examine an alternate approach which gives the same result by a very different method. The method uses capsules to give a zero-knowledge protocol to interactively prove that a given Boolean formula (or arbitrary Boolean circuit with in-degree 2) has a satisfying assignment. Brassard and Crepeau in [BrCr86] independently of both this work and [GMW86] have achieved the same result, and a similar result is given in [Cha86].

The major advantage of this method over the original is efficiency. When a Boolean formula or circuit is reduced to a colorability graph, the number of vertices and edges in the resulting graph is linear in the size of the Boolean formula. Each stage of the interactive proof protocol of Goldreich, Micali, and Wigderson, however, requires a new encryption of the entire graph; and for any fixed confidence level desired, their protocol requires a number of stages which is linear in the number of edges in the graph. Thus, the number of probabilistic encryptions required by this protocol grows quadratically with the size of the graph (or circuit). Because of the local nature of the method presented below, re-encryption is not necessary, and the number of probabilistic encryptions required grows only linearly with the size of the circuit (or graph).

The major disadvantage of this method compared to the original method is that the new procedure requires a (seemingly) stronger cryptographic assumption. Although both methods require a probabilistic encryption function — the best known of which is based on residue classes ([GoMi84]), the method given here requires a probabilistic encryption function for which two encrypted values can be proven (in a zero-knowledge manner) to be encryptions of the same value. Although this property *is* easily achieved by the residue class based probabilistic encryption (Lemma 3), it is not at all obvious that every probabilistic encryption function has this property. However, by observing that the problem of inverting a probabilistic encryption function is itself in $\mathcal{NP}$, the original Goldreich, Micali, and Wigderson result can be applied to show that the cryptographic assumption required here is, in fact, no stronger than the assumption of the existence of an arbitrary probabilistic cryptosystem.

5.1 The Satisfiability Scheme

The basic idea of the scheme is again deceptively simple. If Alice wants to prove to Bob that a given formula is satisfiable (and Alice has a satisfying assignment), Alice begins by choosing an n which is the product of two large primes and providing

Bob with n and a y (with Jacobi symbol 1) which is not a quadratic residue modulo n. This is merely the establishment of a probabilistic encryption function. Alice can convince Bob that y is a non-residue by engaging in the non-residuosity protocol of section 4.2.1 or by choosing n of a special form so that (for instance) $y = -1$ is a non-residue.

Alice then draws a circuit to compute the Boolean function (in the obvious way), selects a satisfying assignment and sends Bob an encryption of this assignment (for each variable, Alice sends Bob a residue if that variable is False/0/Off and a non-residue if that variable is True/1/On). Alice then encrypts the output of each gate of the circuit in the same manner and sends Bob these encrypted values as well.

For each gate in the circuit, Alice then interactively proves to Bob that the gate computes the required function. The computation of an AND gate will be shown here, and other Boolean functions will become apparent.

To prove that a given gate computes an AND on its inputs, a full truth table for AND is used. There are, of course, four possibilities: either both inputs and the output are 0; the first input is 0, the second is 1, and the output is 0; the first input is 1, the second is 0, and the output is 0; or both inputs and the output are 1. A four-component capsule can now be prepared such that each of the four components of the capsule is itself an *ordered* triple. To compute AND, the four (unordered) components of the capsule consist of (ordered) triples whose elements are members of residue classes $(0,0,0)$, $(0,1,0)$, $(1,0,0)$, and $(1,1,1)$. Once a capsule C is interactively proven to be of this form, Alice selects the component which corresponds to the actual input and output values of the gate and proves that they match by releasing a square root of each quotient.

To prove that a capsule C is of the above form, Alice prepares many (say 100) capsules of this form and Bob selects an arbitrary subset to be opened. Alice then proves that each unopened capsule matches C by matching corresponding components and releasing square roots of the quotients of all three elements of each triple to show that they do, in fact, match.

Finally, Alice interactively proves that the output of the circuit is 1 by proving that this value is a non-residue as in section 4.2.1.[2]

Remark Some gates may be computed without the need for an interactive proof. For example, an encrypted value may be complemented simply by multiplying it by y, and the XOR of two or more encrypted values is represented by their product (Lemma 2).

A mechanism which could obviate the need for any interactive proofs to verify gate validity is highly desirable. An encryption homomorphism which allows the direct computation of AND or OR together with NOT would of course suffice, and this would allow satisfiablitiy to be proven with a single interactive proof of

[2]Chaum points out in his work that with a slight modification of this protocol, the need for this final interactive proof can be eliminated.

the value of the output. However, no such probabilistic encryption has yet been found.

6 Conclusions

The method of cryptographic capsules, especially (but not exclusively) when combined with residue classes, seems to be a powerful tool with many applications. This simple tool makes possible several protocols which would be impractical or completely impossible without them. In addition, several previously published protocols can be significantly simplified by the use of capsules.

It is believed that capsules may have many applications which go well beyond those described here, and they may become a standard tool in the design of interactive protocols.

Acknowledgements

The author would like to express many thanks to Oded Goldreich, Shafi Goldwasser, Neil Immerman, Jerry Leichter, Ruben Michel, and David Wittenberg for their help in developing this work and to Mike Fischer who, in addition to giving much guidance and many helpful criticisms, orgininally suggested the use of the term "capsules".

References

[Ben86] **Benaloh, J.** "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[BeYu86] **Benaloh, J.** and **Yung, M.** "Distributing the Power of a Government to Enhance the Privacy of Voters." *Proc.* 5th *ACM Symp. on Principles of Distributed Computing*, Calgary, AB (Aug. 1986), 52–62.

[BrCr86] **Brassard, G.** and **Crepeau, C.** "Zero-Knowledge Simulation of Boolean Circuits." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[Cha86] **Chaum, D.** "Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[CoFi85] **Cohen, J.** and **Fischer, M.** "A Robust and Verifiable Cryptographically Secure Election Scheme." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[Coh86] **Cohen, J.** "Improving Privacy in Cryptographic Elections." *TR-454, Yale University, Departement of Computer Science*, New Haven, CT (Feb. 1986).

[FMR84] **Fischer, M., Micali, S.,** and **Rackoff, C.** "A Secure Protocol for the Oblivious Transfer." Presented at *Eurocrypt84*, Paris, France (Apr. 1984). (Not in proceedings.)

[GHY85] **Galil, Z., Haber, S.,** and **Yung, M.** "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[GoMi84] **Goldwasser, S.** and **Micali, S.** "Probabilistic Encryption." *J. Comput. System Sci. 28*, 2 (Apr. 1984), 270–299.

[GMR85] **Goldwasser, S., Micali, S.,** and **Rackoff, C.** "The Knowledge of Complexity of Interactive Proof-Systems." *Proc.* 17th *ACM Symp. on Theory of Computing*, Providence, RI (May 1985), 291–304.

[GMW86] **Goldreich, O., Micali, S.,** and **Wigderson, A.** "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design." *Proc.* 27th *IEEE Symp. on Foundations of Computer Science*, Toronto, ON (Oct. 1986), 174–187.

[Yao82] **Yao, A.** "Theory and Applications of Trapdoor Functions." *Proc.* 23rd *IEEE Symp. on Foundations of Computer Science*, Chicago, IL (Nov. 1982), 80–91.