PUBLIC QUERY TECHNOLOGY: FEATURES, LIMITATIONS AND AN APPLICATION

Bob Smart [1]

[1] School of Accounting and Information Systems,
University of South Australia

ABSTRACT: Public Query is the name given by the author to a technology that is a combination of cryptography and small scale mobile communications created to enable secure querying of extremely-small databases in ad-hoc wireless peer-to-peer environments. Currently, in the preliminary stage of its development, this technology allows for advanced security features in one-to-many and many-to-many wireless architectures. Various symmetric encryption algorithms, such as Diffie-Hellmann and Elliptic Curve Cryptography (ECC) integrated with optimisation techniques, enable the technology to be implemented in the very simple microcontrollers used in smart cards. Our developed prototype (GoSo) of an application of this technology, and our analysis of that prototype highlights several unique advantages as well as limitations of the technology. These suggest its use in short-range, ad-hoc wireless network applications. The major benefits are repetitive small data transfer made possible by the processing power efficiency, human-language independence and advanced security.

INTRODUCTION

Public Query[1] technology enables confidential database querying within unprotected ad-hoc wireless environments. Wireless network technologies have good technical characteristics for one-to-many communication in peer-to-peer environments, but the security problems of confidentiality, integrity and authentication often prevents its safe implementation [Sm03]. Public Query technology is designed to overcome these limitations and establish a secure serverless communication environment for wireless devices, without established trust.

This paper, which will present details and analysis of a software prototype application of the Public Query technology, will point out the prototype's key features, its practical advantages over conventional mobile client-server systems and its limitations. Finally, it will suggest the direction for further research and development.

The number of practical applications for Public Query technology are numerous [Sm03]. Thus include in logistics, marketing and telematics. They may be used as a companion to 3G mobile telephony services, although the technology was primarily designed to work independently in areas not suitable for 3G. One application, not developed in this paper, might be used to match a commuter driver who is passing a hitchhiker in secure manner by means of a simple automatic wireless communication between the two. They could communicate directly and independently without any service provider. They could set their security level independently, which is required in environment without central server and because the hitchhiker would be communicating their desired destination with all the nearby drivers simultaneously. The hitchhiker may never know the driver's route and driver may never know the hitchhiker's destination, unless they have a matching part of route or a 'keyword' in their wireless communication, which both have agreed to set in their wireless communicator. Public Query is human-language independent, which provides great opportunities in tourism and other global industries, however, the application presented here is for a match-mating communication called GoSo.

Prototype Application Description

The object of building the prototype was the creation of virtual electronic mate-finders, internally called GoSo – (Go Socialise!). This application has proved to be a good platform, both for development and

[1] The name is given by author

to appreciate other examples and applications for Public Query. This application requires object organisation and the need for confidential data transfer between specific, but not previously specified groups. In our mate-finding application GoSo every human individual is first represented as a complex object. These are a combination of confidential and public information representing feelings, features, styles, tastes and experiences that are formatted as numbers, words, ranges, groups and other complex data types. These are then select by the broadcasting person in order to start the process of matching people. The immediate problem is one of secrecy given these objects are to be communicated to nearby people with the appropriate technology. The intend is for say a girl to broadcast for other girls only with the same feature objects highlighted, such as a preferred movie. When broadcasting for matching girls, girls who have not indicated an interest in that movie, or boys who want to be noticed should not be aware of the broadcasters requests nor be able to respond to her queries anyway. The encryption model and blind response query technology will protect her and receivers from every broadcast except the matching one. All other communications are treated as mischievous spam, ignored and deleted automatically. Only matched feature objects are brought to the attention of the two communicators.

RESEARCH METHOD

This prototype application of the Public Query technology has been develop and simulated to date only in an pc environment, this enabled the architecture related problems to be more carefully defined and analysed together with review of database security controls and applied cryptography solutions for low power processor systems. The research method chosen was rapid prototyping, developed by agile programming technique. A MS Windows based prototype designed using Visual Basic, provided foundation for this paper, together with relevant literature review.

The GoSo prototyping software simulated internal processes and communication between two emulated prototype devices that use Public Query technologies. Prototype development was performed using the GoSo development platform, called Smartist. Smartist enables fast and easy modification of GoSo models during prototyping. It was designed in Microsoft Visual Basic and uses a MS Access database.

Object Database:

Being an object based system is problematic for the relational database nature of MS Access. For this purpose, an Entity Relation Diagram (ERD) was created to simulate an object database structure with properties and methods [SH03]. The ERD diagram is shown in Appendix 1. Although, the system is not compliant with Object Database Management Group (ODMG) standards, Smartist can construct or use Object Definition Language (ODL) code which GoSo uses internally. See Hoffer et al [HPM02] and Cattell et al. [Cat00] for more thorough coverage of object database modelling.

INNOVATIVE TECHNOLOGY FEATURES

The prototype developed has features which to the best of our knowledge are unique, or at least have not been used before for this purpose. Therefore they require a public discussion and further testing.

Blind Response

Blind response is defined as when the answering entity answers the query without understanding its meaning. It processes the encrypted query, called a hook, through data as a kind of filter and a result is returned. Confidentiality is preserved trough non-retention of the transaction by the system. Only a querying entity could understand both the meaning of the query and the answer as well. At the same time, if the answering party has sent non-required information, the querying party would not be able to understand the meaning of non-required information. This protects the querying party from deliberate noise, such is spam, because the answering party is aware that only the required information would be understood.

The main problem with this approach is noise because an answering party can respond with a significant amount of data, and the querying party has to filter out non-required from the required data. Whilst distinguishing what was ‘asked for’ from non-required data is efficient process for the querying party, problem lies into large amount of data-transfer, especially in many-to-one and many-to-many

architecture. A trade-off exists between confidentiality and data transfer amount. The GoSo prototype has been designed to allow the user to determine the level of trade-off. The querying party can send additional information with a query that informs answering parties which category of data objects to process, or not to answer at all, if they do not have all required properties in database. The answering party also can determine the minimum level of security required to respond. With this approach both parties actively determined the security standard for their communication. This level is determined by domain size (the level of 'generality' of request), number of available answering parties or specified by the database category. This approach could save processing time as well as confidentiality.

For example the answering party may decide not to answer the question: "What is your favorite reading". This query requires processing of all subclasses of reading class and would have to many responses. On the other hand the question, "What is your favourite magazine", might be precise enough to produce an efficient number of responses and maintain the deserved level of privacy.

It is important to reiterate that the answering party does not 'understand' questions, but in the above example - it knows that query processing is limited to the general printed textual domain or printed magazines sub-domain. The resolution level determines whether the query is acceptable or not.

Public Query Object Encryption

Most encryption technologies scramble alphanumeric messages, audio or video. Until recently, object encryption was not treated separately because it was perceived that every object could be represented as multimedia information. On the other hand, the internal organization of an object could be used in the encryption process and thus sometimes provides increased security as it provides data compression as well. Our approach is that an object class or a blueprint is public and not confidential, but object settings for every object are secret and communicated in encrypted form. The main advantage still lies in flexible and independent settings for confidentiality, authenticity and integrity. One of the basic rules of security in computing is that computer items must be protected to a degree consistent with their value [PP03]. Public query object encryption provides transparent adjustment of security characteristics for every single object in transmission.

Interest in the object encryption is growing and hopefully it will be used in some form in the future. The Internet Engineering Task Force – IETF, is currently creating drafts for End-to-End Object Encryption in XMPP [SA03], but their principles and aims are different from the Public Query model used in GoSo application.

Abstract Object Types

Information stored in GoSo database is meaning oriented. It is organized through objects that have complex types related to directory services. In our GoSo matchmaking example, objects that create a person have abstract types. Although in communication we represent feelings with words (strings) like happy, sad, angry, they are not a type of string. These words are just English Language Textual Interface properties. For our system, they are a 'feeling' abstract object type with its own weight, significance and impact. This might lead to machines that seem sensible and intelligent.

Keyless Encryption Algorithm

The encryption algorithm implemented in the Public Query technology is internally called SM-102 (Secure Meeting Version 1/ Year 02). It does not use encryption and decryption keys in its processes. Although there are other existing keyless algorithms, this concept takes a new approach. The equivalent of a key is created from object properties and random numbers, but because every single piece of information is transferred with different random value and has different properties, there is no uniform key in any stage of the information transfer. This solution in which "plainobjects" dynamically change encryption key is a concept used in the German Enigma machine where key was changed after every new letter of plaintext [Sin99].

Encryption Principle:

The encryption algorithm SM-102 has three inputs in the encryption process:

- An identifier for object properties/schema

- A random number
- A time stamp

The result is an encrypted query called a Hook.

The answering party also uses three inputs similar to the query except that a timestamp is replaced with the hook. The inputs are

- An identifier for object properties/schema
- A random number
- A hook

The math behind SM102 relies on the Diffie-Hellman [DHB77] key exchange algorithm which uses a non reversible process to communicate a single random value between two parties. The developer modified the algorithm to allow check if another party holds particular value in the manner that cannot be reversed. In the basic Diffie-Hellman equation, $\mathbf{y = g^x \% p}$, [AS01] the hook is created with the time stamp as **g**, the random number as **x** and the identifier for object properties/schema as **p**, which has to be a prime number. The produced value **y** is the hook. The answering party use the hook instead of the time stamp (**g**), with its own **x** and **p** values. At the same time it uses the time stamp as **g**, with **x** and **p** values to create its response-hook, which is posted together with the answer. Querying party process the response-hook in the same equation and compare that value with the answer[2]. The result can be true or false, or one and zero, but the additional logic and object structure could provide standard data transfer capabilities. All rules for selection of g, x and p values are inherited from the Diffie-Hellman algorithm [DHB77]. The **p** value is the secret value or an object property known only to the querying and answering party if they have a match (true value).

RELATED TECHNOLOGIES IMPLEMENTED IN PROTOTYPE

Public Query technology performed on application and presentation OSI[3] layers is basically a two-step process: Sending the query and receiving the answer. This simplicity is important for traffic efficiency because every additional communication may be multiplied by the number of participants.

Managing the network layer handshake and maintaining ad-hoc networks with device within the range is handled by various communication standards such are IEEE 802.15.4 and IEEE 802.11x.This will not be discussed further as it is out of scope of this paper. The standards also manage multiple simultaneous signals sent in many-to-many or many-to-one structures as well as the session management process.

Naming and Trading Services

Peer-to-peer networking with unknown parties requires precise object classification and definitions with numeric identifiers. This common collaboration problem [Coy02] is addressed within UDDI[4] [Ram02], and CORBA[5] [OMG03] naming and trading services which are electronic equivalents of information directories like the Yellow Pages [Wh99]. The same services can be used for public query technology, but additional information and object class identifiers in specific formats would be required.

Encryption Model

Public Query technology can use any key exchange algorithm such as Diffie-Helman or Elliptic Curve Cryptography. In the GoSo prototype we use a symmetric encryption process based on the Diffie-Helman [DHB77] key management algorithm. Although this algorithm is widely accepted and acknowledged as a secure and reliable, this does not prove that the resulting encryption algorithm is

---

[2] See Appendix 2 for SM102 example
[3] Open Systems Interconnection
[4] Universal Discovery, Description And Integration Of Web Services
[5] Common Object Request Broker Architecture

cryptanalysis-proof. Biham and Shamir [BS90] have proven that sometimes even the slightest modification can significantly weaken the algorithm. Therefore, differential cryptanalysis is planned for the evaluation process in coming months. Results of this analysis are not crucial because system does not rely on this particular algorithm and if proven necessary, similar algorithms like Elliptic Curve Cryptography [Sch96] can be used.

Known Encryption Weaknesses:

The encryption model used for public querying in GoSo has known weaknesses, which could be used by cryptanalysts. Every query and database object is processed with large, but finite group of random numbers. In addition, every property has a limited number of values. These numbers are multiplied and that creates a possible number of message values. Although this number is extremely large, it leaves a group of impossible values which provide an opportunity for cryptanalysts to identify what is not a message. Thus, theoretically they could deduce what is a message.

This deduction process is easy for messages and object properties with small number of possible values. For instance: if there are only two possible values and someone proves that one of them is not selected, the selected one is discovered as well. However, calculating that impossible value requires a lot of processing power although the encryption is easy to perform, but difficult to invert [DHB77]. In addition, the encryption process is done in iterations, which increase the security level. Faster processors mean less impossible message values and therefore better security. Careful selection of random numbers could significantly improve security on low processor powered systems, but this theory of numbers issue is also out of the scope of this paper.

Distributed Processing Power

Encrypting and decrypting are well known as mathematically intensive operations. For a wireless, battery powered device this is a significant problem [Max02]. The GoSo prototype uses a distributed processing power technique, which enables wireless devices to use desktop computer processors to compile in a manner similar to a code compiling process. Software for every device, which uses public query technology, is created on a desktop computer and optimised for queries, supported by the device. Complex mathematic operations are performed on fully powered PC computers. Then, optimised and self-sufficient software is loaded into a wireless device where these objects are additionally processed only with random numbers and time stamps. This mobile part of public query technology use simple add and subtract operations which are reasonably easy to perform even on low powered processors which run at 4MHz clock or more. In addition, SM-102 is integer based which makes it suitable for Java 2 ME CLDC[6] configuration which does not support floating point operations. This algorithm also overcomes the need for using limited Sandbox security model implemented in Java 2ME [HO02].

RESULTING FEATURES

Asymmetric Processing Power Compatibility

Compatibility of two devices with significantly different processing powers and maximum encryption strength for each of them is preserved. This is performed with iterative steps designed so that new iteration adds encryption strength, but remains compatible for another party. As previously stated, a party that answers the query does not understand the meaning, rather just process the query with their data. The important requirement is for the query to remain in a compatible format. From a cryptographic view, the answering party just adds a new protection layer around the query and response. Even if this security layer is created on a much faster processor, the querying party with slow processor can easily decrypt it.

Time Stamp

The Time stamp is an important mechanism, which prevents active wiretapping [PP03]. Although, queries are public, in some circumstances they can trigger objects' methods or be used for

[6] Java 2 Micro Edition – Connected, Limited Devices Configuration

authentication. If someone records a broadcasted hook, it can be resent later to gain certain privileges. The Timestamp makes this impossible, because the query will be badly processed and unusable. A time stamp is used as one parameter for the creation of hook and it is also sent in plaintext form together with hook. If there is a big difference in the system time between devices involved, the answering device may ask to be sent the same query with the answering device system time stamp. In addition to time stamp, only the hook sender could understand answer to posted query. This feature prevents session hijacking [Vin02].

Database Security

Public Query technology is designed as a system for secure database querying. Every data object must have defined access status as well as suppression and concealing levels which protect from inference – the obtaining of sensitive data from non-sensitive data [PP03]. Denning and Schlorer [DS83] define concealing as providing close, but not exact data by use of rounding or value ranges. Suppression represents hiding of sensitive data. Public Query technology allows a use of different range levels to conceal responses as well as to limit and control a number of queries from user per object.

RESULTS

The GoSo prototype has proved technology efficiency. An application and presentation layer with database management system for a GoSo device created a 14kb ActiveX object. The database used in tests had 9 object-properties with an average of 7 values per property. The database binary size was 2814 bytes. Query size was 36 bytes and answer was 74 bytes long at included all 9 object-properties. All properties were initially shared by both devices. This result is the best scenario and in the case of non-supported object-properties our estimation is that it will need in average 4 times more data then “plainobjects”, which makes the system inefficient for large databases.

Communication did not include checksums, hashes, entity and session identifiers. This additional data is determined by the communication standard and it is not within the scope of this research. In addition, queried data size-to-data transfer ratio can significantly vary due to object abstract type and properties structure and number of values.

CONCLUSION

The GoSo project created many questions, with most of them addressed and resulting problems solved during development. Nevertheless, it highlighted numerous system limitations, mainly regarding database size. The GoSo system introduced exciting novel features such is blind response querying. As a system designed with security and privacy in mind it has a broad range of possible applications. The mate-finding system proved during testing, to be a candidate for commercial development with minimal corrections. In addition, the development platform GoSo/Smartist is invaluable resource for further research, but it needs improvements.

The GoSo prototype provides for high memory and data transfer efficiency in simulated ideal circumstances only while using micro-databases. Whilst the key feature is the blind response querying aspect, language independence, simplicity of use and low processor requirements makes it a perfect companion for battery-powered mobile devices with limited display and input capabilities.

Although it theoretically might be used for large Internet databases, there are already better and more efficient, secure technologies available. To use standard Internet databases as well, the following limitations have to be overcome:

Unknown Parties Handling

Public query technology whilst designed for ad-hoc, peer-to-peer wireless networks with unknown parties should be perceived provisionally. Because encryption procedure involves expected object properties/schemas, querying parties cannot fully communicate with unknown type of parties. Once this function is developed, if the parties’ schemas are not already shared, they would negotiate exchange of database schemas and object classes.

Unreliability

Freedom and flexibility can lead to anarchy and this is a problem for Public Query technology. Complex internal query and answer design, which involves three input parameters, may create unreliability issues. For example, it is theoretically possible for a not required response, with inadequate database properties and a bad time stamp to turn into valid information at the final check stage because multiple errors could negate each other. This is a very uncommon scenario, but further plans involve calculation of error probability and the design of additional controls, which will prevent this threat.

Limited number of queries

Both, the distributed processing power approach and a simplified interface limit the system to preprogrammed queries. Faster systems with more complex interfaces could allow for the creation of ad-hoc queries. The idea is to allow users to independently create and choose queries, process them and then load the application onto a mobile device. There are other possible solutions to this problem, but they involve potential security risks, increased complexity of use, higher energy consumption and more expensive devices.

In addition, the blind response feature does no allow for the use of grouping functions in database queries. This severely limits functionality but prevents attacks on confidential information through the combined results.

FURTHER RESEARCH AND DEVELOPMENT

Security algorithm SM-102 requires independent evaluation and differential cryptanalysis. As every security system it also needs usage tests.

The GoSo prototype worked in restricted and ideal virtual conditions. It was tested in simulated 1-to-1 architecture. Further development includes the creation of Smartist plug-ins for Java and MS .Net GoSo code. These plug-ins should combine the prototype solutions and framework security standards provided with the development tools [LMP02]. The next phase includes real life tests on Java/Bluetooth mobile phones, which support up to 8 devices in network [Gal02] and Zigbee IEEE802.15.4 based networks with up to 127 devices [ZB03] which may provide more objective results.

ACKNOWLEDGEMENTS

It has been a great honour to have Sam Horrocks and Mike Metcalfe as my editors, as well as Adam Jenkins which has provided me with sources and invaluable programming skills.

REFERENCES

[AS01] A. Seth, *Data Encryption Page: Diffie-Hellman*, Available from Internet <http://www.anujseth.com/crypto/keyexch/dh.html>, 2002.

[BS90] E, Biham, and A. Shamir, Differential Cryptoanalysis of DES-like Cryptosystems, *Proceedings at CRYPTO '90 Conference*, 1990.

[Cat00] R. Cattellm, *Object Data Management: Object-Oriented and Extended Relational Database Systems*, Addisson-Wesley, Massachusetts, USA, 1994.

[Coy02] F. Coyle, *XML, Web Services, and the Data Revolution*, Addison-Wesley Information Technology Series, Boston, USA, 2002.

[DS83] D. Denning & J. Schlorer, Interface Controls of Statistical Data Bases, *IEEE Computer*, Volume 16 No7, July 1983, pp.69-82, 1983.

[DHB77] M. Diffie, B. Hellman, and M. Ralph, *Cryptographic apparatus and method*, US Patent 4200770, Available from Internet <http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=/netahtml/search-bool.html&r=1&f=G&l=50&co1=AND&d=ptxt&s1=4200770.WKU.&OS=PN/4200770&RS=PN/4200770>, 1977.

[Gal02] N. Galbreath, *Cryptography for Internet and database applications: developing secret and public key techniques with Java*, John Wiley & Sons, London, UK, 2002.

[HO02] I. Haque, B. O'Connor, *Professional Mindware: J2ME Enterprise Development,* M & T Books, New York, USA, 2002.

[HPM02] J. Hoffer, M, Prescott & F. McFadden, *Modern Database Management*, 6th international edition, Prentice Hall, Upper Saddle River, New Jersey, USA, 2002.

[LMP02] B. La Macchia, S. Lange, M. Lyons, R. Martin, K., Price K, *NET: Framework Security*, Addison-Wesley, Indianapolis, USA, 2002.

[Max02] M. Maxim, *Wireless security*, McGraw-Hill Osborne Media, Emeryville, USA, 2002.

[OMG03] Object Management Group, *CORBA FAQ*, www.omg.org, Available from Internet <http://www.omg.org/gettingstarted/corbafaq.htm>, 2003.

[PP03] C. Pfleeger & S. Pfleeger, *Security in Computing, 3rd Edition, Prentice Hall Professional Technical Reference*, Upper Sadle River, New Jersey, 2003.

[Ram02] Rambhia, A. 2002, *XML: Distributed Systems Design*, Sams, Indianapolis, USA.

[SA03] P. Saint-Andre, *End-to-End Object Encryption in XMPP*, The Internet Engineering Task Force, Available on Internet <http://www.tieke.fi/ietf-drafts.nsf/0/365AE04F6B84768CC2256D2E00419B80?OpenDocument>, 2003.

[Sch96] B. Schneier, Applied Cryptography: protocols, algorithms, and source code in, 2nd edition, Wiley, New York, USA,1996.

[Sin99] S. Singh, The code book: the science of secrecy from Mary Queen of Scots to quantum cryptography, Doubleday, New York, USA, 1999.

[SH03] J. Smith & A. Hodgett, Database Design: SQL Notes, University of South Australia, Adelaide, South Australia, Australia 2003.

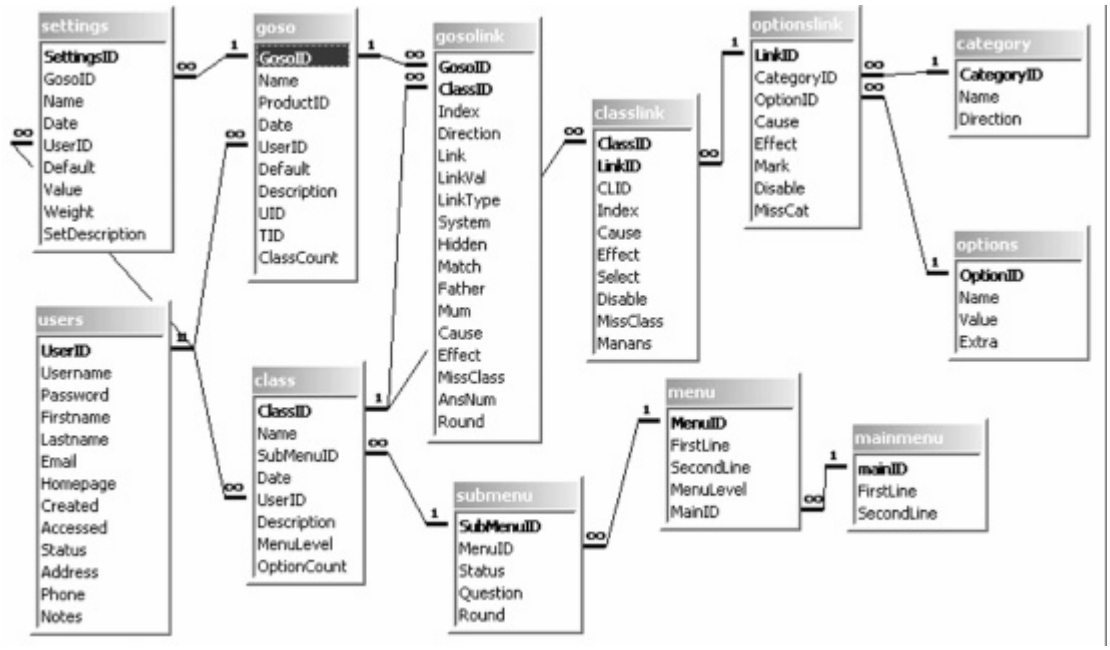
[Sm03] B. Smart, Technical And Commercial Aspects Of Secure Wireless P2P Networks With Unknown Entities, self published, Available on Internet <http://home.iprimus.com.au/bobioso/BobWirelessP2PAUSCC.pdf>

[Vin02] R.D. Vines, *Wireless security essentials: defending mobile systems from data piracy*, John Wiley & Sons, London, UK, 2002.

[ZB03] *The Official Website of ZigBee Alliance*, ZigBee Alliance, Global Inventures, Available from Internet <http://www.zigbee.com>, 2003.

APPENDIX 1: ERD

Pic 1. ERD for Access database which emulates object structure



APPENDIX 2: SM102 Algorithm Principle

$$\mathbf{Y = G^X \% P}$$

Selected timestamp (G) is 5 in both examples. This is the only public parameter which is negotiated.
Matching Values

Q is querying if A has number 11 stored Q chooses 2 for random number	A has number **11** stored A choose 3 for random number
5^2mod11=25mod11=3	5^3mod11=125mod11=4
	3^3mod11=27mod11=5
If 4^2mod11=5 Then True	**TRUE**

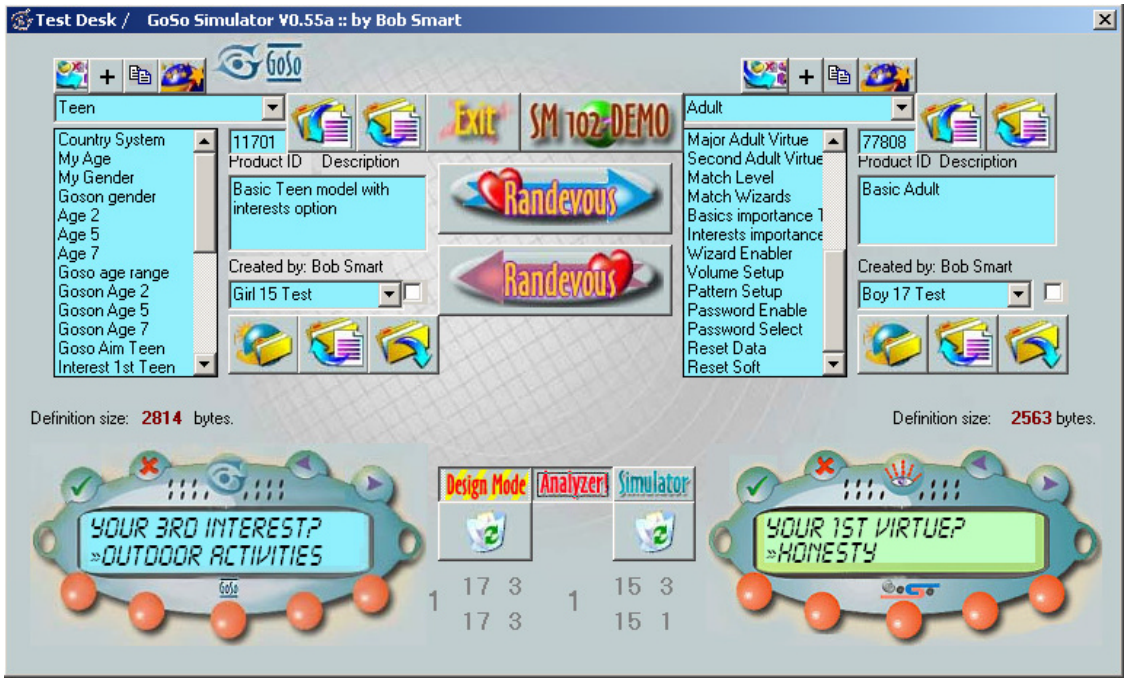
Protocol: Q sends the hook = 3, together with domain and timestamp in plain text. A responds with concatenated values 4 and 5 (45). Only Q party from answered number 45 can get a secret value **11**.

Non-Matching Values

Q is querying if A has number 11 stored Q choose 2 for random number	A has number **13** stored A choose 3 for random number
5^2mod11=25mod11=3	5^3mod13=125mod13=8
	3^3mod13=27mod13=1
If 8^2mod11=1 Then True (the result is 9)	**FALSE**

In this case the answer is 81 and Q party only knows that A value is NOT 11. The potential problem is if the result of any equation on the A-side is 12 (mod 13 allows that), thus Q would know that A has a number larger then 12. To overcome this problem, the hooks and answers only less then the smallest possible identifier (P) value are allowed.
This is just a basic example. In order to be reliable and secure, much larger prime numbers should be used. Another rules involve: **P** is a prime number larger then 2. **G** is any integer less then P. **X** is any integer larger then 1 and less then P-1 [DHB77].

APPENDIX 3: Screen Snapshots
Pic 2. GoSo simulation/test screen



Pic 2: GoSo Designer in Smartist